# A DEEP IMAGE CLASSIFICATION APPROACH FOR FISHING ACTIVITY DETECTION FROM AIS DATA

by

Gashin Ghazizadeh

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2019

*I dedicate this thesis to my family whose love always shows me the way.*

# Table of Contents

# List of Tables

# List of Figures

## Abstract

Maritime transport and vessel activities on the ocean have a significant impact on marine life, which consequently affects human life. Therefore analyzing and monitoring the fishing activities using the vast amount of the data that Satellite-based Automatic Information Systems (S-AIS) provides, using machine learning methods, has become more popular than before. Most of the works on S-AIS data try to detect fishing activities using point-based methods that require significant preprocessing steps to extract meaningful features for the samples. In this work, we use a different perspective toward trajectory data. Human brain cannot understand the type of activity by looking at the point-based data, while experts can easily recognize fishing activity by looking at the movement of a ship on the map. In addition, the significant advances in the field of computer vision made us convert the problem to an image classification task. Informative parts of the trajectories that can contain points where the vessels have been doing an activity are extracted as sub-trajectories using DBSCAN. These sub-trajectories are depicted by drawing the lines between points and saved as images. With the new created image dataset, different CNN models are trained. Our method, unlike other methods, does not need prior information about the movement and can be used for all types of fishing vessels. Our results on the images created from the trajectories of different regions of the world show excellent performance that can be applied for detecting fishing activity from trajectory data.

## List of Abbreviations Used

| | |
|---|---|
| AIS | Automatic Information Systems. |
| ANN | Artificial Neural Network. |
| CNN | Convolutional Neural Network. |
| DBSCAN | Density-based spatial clustering of applications with noise. |
| DT | Decision Tree. |
| DTW | Dynamic Time Warping. |
| GAN | Generative Adversarial Networks. |
| GPS | Global Positioning System. |
| HCR | Heading Change Rate. |
| HDOP | Horizontal Dilution Of Precision. |
| IMO | International Maritime Organization. |
| LR | Learning Rate. |
| MMSI | Maritime Mobile Service Identity. |
| NLP | Natural Language Processing. |
| NN | Neural Network. |
| PCA | Principal Component Analysis. |
| RF | Random Forest. |

| | |
|---|---|
| RNN | Recurrent Neural Network. |
| S-AIS | Satellite-based Automatic Information Systems. |
| SOG | Speed Over Ground. |
| SR | Stop Rate. |
| SVM | Support Vector Machine. |
| VMS | Vessel Monitoring System. |

## Acknowledgements

I consider myself blessed to have got an opportunity to study under Dr. Stan Matwin's supervision, who has been highly supportive and encouraging since the beginning. He always helped me through the ups and downs of my research with his insightful ideas and directions. I am ever grateful to him for funding and supporting my entire master's program.

I am grateful to my teachers – Dr. Stan Matwin, Dr. Thomas Trappenberg, Dr. Robert Beiko, and Dr. Stephen Brooks – for teaching Machine Learning, Bioinformatics Algorithms and Visualization so well! The ideas I got from these classes helped me to decide on my work later.

I would like to thank Dr. Vlado Kesselj for giving me an opportunity to work with him and his students on a project and guide me through it. I got exposed to a lot of interesting ideas in NLP and helped me in getting a Co-op experience.

My sincere thanks also go to Dr. Omid Isfahani, who helped me with his insightful information. Without his kind input, this thesis would have taken a long time.

Finally, I must express my very profound gratitude to my parents, my brother, and my partner –Erfan Gheibi–, whose love and guidance are with me in whatever I pursue. This accomplishment would not have been possible without them and their unfailing support and continuous encouragement. Thank you.

# Chapter 1

# Introduction

The primary objective of this thesis is detecting fishing activities from the trajectory data that is given for vessels in different regions of the world. In order to define our problem in more detail, the importance of this task and an introduction to trajectory data related tasks has been given.

## 1.1 Fishing Activity Detection

From the very beginning, one of the main sources of human food has been seafood. In the last century, with the rapid growth of technology and the tools humans use, fishing has become an easy and affordable way of providing food. In certain areas of the world, fishing has grown so fast that certain species of the fish and have been either underpopulated or depleted. Overfishing can cause serious problems to the ecosystem; hence, certain rules have been put together to prevent it from happening.

Despite all the efforts that have been made to reduce overfishing, illegal fishing is one of the major reasons for several fish species' depletion. It has been estimated that 30% of total fishing catches are done illegally [14]; therefore, detecting illegal fishing activities is an important problem that needs to be solved.

Different ways of observing marine activities from tower-based approaches, to Satellite-based Automatic Identification System (S-AIS) that monitor ships all around the world, have been used during years to detect fishing activities. Experts can tell the difference between fishing versus non-fishing activities by monitoring ship movements in the middle of the oceans. While these monitoring systems can effectively address the issue of fishing detection, their scalability can cause problems as the amount of data that AIS can provide for the experts is enormous, and the process is done manually.

Furthermore, detecting fishing activities from AIS data is important from a scientific perspective. This task is in high correlation with many other tasks that work with different trajectory datasets; therefore, creating a method that can work on this type of data will help

to solve many more tasks.

Our work tackles the problem of fishing detection by the use of AIS trajectory data. Our main contribution is automating fishing detection task using machine learning methods. The uniqueness of our work is the data transformation idea, which converts spatiotemporal data to an image dataset. Our model then solves the fishing detection problem as a image classification task from the created dataset.

## 1.2 Problem Definition

The progress in technology in the last decade has enabled us to make tools to collect the data of mobile objects, humans, and animals. The data that is collected is called Trajectory Data. Trajectory data is spatiotemporal data that is made of a sequence of geolocations in time that can make a trace of the movement of an object [1]. For example, cell phones store their location whenever they are connected to the Global System for Mobile Communication (GSM) network; therefore, a digital trace of the people can be made using their sequence of locations in time. As a result, the data that is collected from cell phones is trajectory data.

Based on the definition of trajectory data, we can create images of the movement of an object. The points that an object traverses during time, which make the trajectory data, can be depicted on a map to show the movement of the object. These image dataset can be used to identify a special type of movement. The problem we try to solve in this thesis is to train a model to be able to identify fishing activity movement of ships by depicting the trajectory data that we have collected. Therefore a computer vision approach for a trajectory classification task is our main goal.

From the beginning, different data mining tasks have been proposed for trajectory datasets. These tasks range from classification and pattern mining to anomaly detection. Trajectory datasets also need several preprocessing steps, including but not limited to noise filtering and segmentation. Many applications need to extract features like speed and acceleration to analyze the data. Also, there are different categories of trajectory data based on the type of tracked movement; for example movement of humans is different from animals and ships. As a result, there are so many different works that have been done on trajectory datasets, which can be summarized in Fig. 1.1 [1].

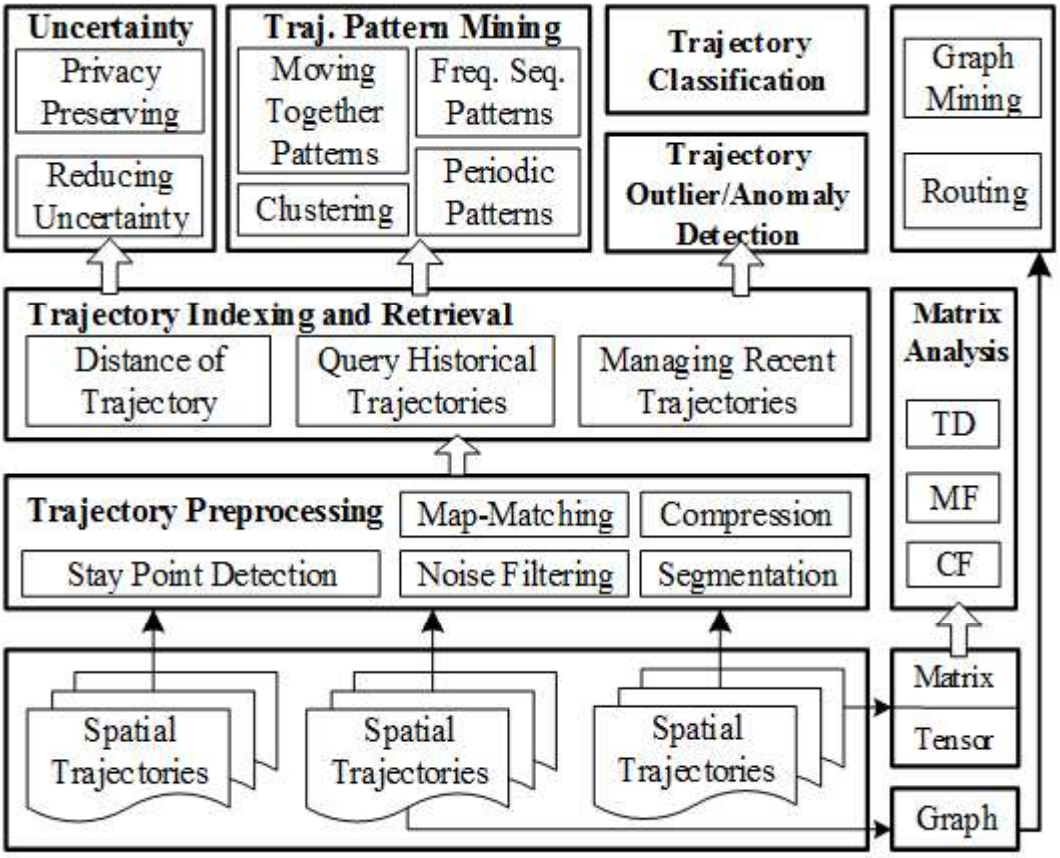



Figure 1.1: Trajectory data related tasks have different steps and purposes that can be summarized into different levels. [1].

Our method requires several of the data mining steps that are shown in Fig. 1.1. A combination of **Noise Filtering**, **Segmentation**, **Trajectory Pattern Mining**, and **Trajectory Classification** have been done in our work. These concepts are explained as follows [1]:

- Noise Filtering: Trajectory data is not always clean, and many points have inaccurate recorded locations due to the sensor noise and other factors. There are several different methods for dealing with noise in trajectory data, that can detect the noise point and either remove it or correct its location.

- Segmentation: A trajectory trace contains different segments that can be separated based on the value of information they contain. For example, in our method, points that are recorded while the vessel is simply moving toward a destination are less critical than the ones recorded while fishing. Therefore, different algorithms have been applied to divide a trajectory trace to smaller segments, which are categorized

as segmentation methods.

- Trajectory Pattern Mining: Depending on the use case, different patterns can be found in groups of trajectories. Patterns that can be extracted from trajectory data range from a number of moving objects that go in the same patterns at the same time like birds, or human parades to objects that have the same type of pattern while moving like different fishing vessels of a single type. Different clustering and mapping methods have been proposed to extract these patterns that fall under pattern mining.

- Trajectory Classification: The task of giving labels to trajectories or smaller segments of them is a classification task. This task usually contains segmentation tasks inside of it, as giving labels to the whole trace of trajectories is not always meaningful. Trajectory classification usually gives semantic meaning to the data.

How these steps have been combined to form our model is explained thoroughly later in Chapter 4. Another aspect of our works that needs to be introduced is using Computer Vision methods.

Computer vision has been one of the fields that has fascinated scientists for a long time because of the human interest in building systems that can see and understand. Automating the tasks that the human visual system can do is one of the main goals of computer vision, and it aims to do so by processing and analyzing digital video and images [15, 16, 17]. Due to the great progress in this field by the use of convolutions neural networks, most of the image classification problems are solved with high accuracy. Hence, we decided to convert our task to an image classification task to create a system to detect fishing activities the way humans do.

This work tries to create a new model that can detect fishing activity from a series of points of trajectory data. AIS can provide vast amounts of trajectory data of ships in different regions of the world. The data contains some features of the movements as well as the location of the vessels on the ocean. This dataset is used to create images from the informative parts of the trajectories that ships are doing an activity. Separating the informative parts of the trajectories that contain points of the vessel during activities is done using clustering and regression algorithms. Subsequently, these images are fed into a Convolutional Neural Network (ConvNet/CNN) for training. The trained CNN can be used to classify new unseen images of raw trajectory data to detect fishing activities.

## 1.3 Questions and Contribution

During years, several different attempts have been made to detect fishing activities from the AIS data, and different machine learning methods have been developed to address the problem; however most of the methods only try to detect fishing activities of several specific vessel types and cannot be generalized. Besides, these methods depend on extracting characteristics of the movements that can only be helpful for certain types of vessels.

On the other hand, all of the previous attempts to detect fishing activities from AIS data automatically, use AIS trajectory data points as input for their method. Afterward, based on the mathematical operations and features that are extracted from the data, the model is trained, whereas, in our work, the model is trained on images. The way experts can tell the difference between fishing, and non-fishing activities is by looking at the movement of the ships. We have used this idea to build a system that can detect fishing activities by looking at the images created from the ships' movements. This approach is general and does not require prior information about the type of ships and their fishing movements to detect them. Using CNNs can help us make our system independent of the vessel type, and achieve high accuracy like other image classification tasks.

The questions that this work tries to answer are:

1. How to detect illegal fishing from AIS data?
2. Can representing trajectories as images improve fishing detection?
3. How does the performance of our proposed method compare to the other individual state-of-the-art classical approaches?
4. Can our method outperform other individual state-of-the-art classical approaches for all types of vessels?

The research summary and contributions of this work can be listed as follows:

- We propose a method for solving fishing detection problem from AIS data by creating images of ship activities and train a model on them. The performance of the model is not dependent on the type of vessel or features that are extracted manually to improve accuracy. The whole process is unsupervised and can be applied to similar tasks.

- A new method for removing non-informative segments from DBSCAN results on fishing trajectories has been implemented. This method will decrease the amount of noise data in the created image dataset. Consequently, the performance of the model will be improved.

- A different perspective for solving problems that humans deal with using their visual system is proposed. These types of problems can be converted to different computer vision tasks like image classification, segmentation, or object detection. This type of data representation can improve the results due to the great progress in the field of computer vision.

- Lastly, a new dataset of images from AIS trajectories for fishing and non-fishing activities has been created, which can be used for further studies regarding the fishing detection problem.

The rest of this thesis is organized as follows: Chapter 2 briefly goes over all of the previous works that have been done. The preliminaries are explained in Chapter 3. Chapter 4 and Chapter 5 explain the proposed method and analyze the results. Lastly, conclusion and future work are given in Chapter 6.

# Chapter 2

# Related Work

With the availability of big amounts of mobility data that have a better quality than before, many researchers have tried to work with trajectory datasets and solve the raised tasks. These tasks range from classification to pattern detection and clustering tasks. In this chapter, we briefly go over the related research papers that have been proposed during the years for both maritime and other types of trajectory data tasks. We divide these works into three categories: Segmentation Methods, Non-maritime Trajectory Classification Research Works, Maritime Trajectory Classification Research Works.

## 2.1 Segmentation Methods

In this section, previous works that have focused on the trajectory segmentation, are explained. One of the steps in our method is segmentation of the data; therefore, these works are related to our work. One of the main uses of trajectory segmentation methods are is to detect stay points in the motion of objects, but in our task, the segmentation methods are used for activity detection.

Several different methods have been proposed for segmentation that are based on DBSCAN. These methods are unsupervised as a result of DBSCAN's nature. T-DBSCAN is a method that was introduced for this purpose [18], which is different from DBSCAN by putting a limit on the number of points in the Eps neighborhood. Based on the definitions they give for stop and move segments, they put certain time limits on the number of points in the Eps neighborhood, so that all the points are not counted. Another method that uses DBSCAN for segmentation was proposed in "An improved DBSCAN algorithm to detect stops in individual trajectories" [19]. This method uses DBSCAN on trajectory data with a new definition for core points. A core point in their work is defined as a point that has a density in its neighborhood higher than a threshold. For calculating their threshold, they use "elbow point" concept from [20]. Although the improvement of these algorithms, increases the accuracy of Stay point detection, they are not suitable for activity detection. Activity

segments can have a larger spatial area than stay points. Therefore defining conditions on density for a small area will lead to poor performance.

CB-SMOT is another method based on DBSCAN that gives an updated definition for core point [21]. In their work, they take the speed of moving objects into consideration by adding time to the conditions of being a core point. They calculate the distance of other points on the trajectory from a point, and if less than Eps, they are considered neighbors. They call this neighborhood the Eps-linear-neighborhood of points. Then they calculate the duration of the neighborhood as the time difference between the first and last points of it. A core point in their method is a point that the duration of its neighborhood is higher than a minimal time.

Other methods mainly focus on characteristics like speed, distance, direction, and time duration between points to address the issue. One of these methods is TRACLUS [22], which tries to divide trajectories based on the distances between points. This method method uses the minimum description length (MDL) principle to find the best points, which they call characteristic points, for separating the different parts. The ideal case of their separation is to find the points where there is a significant change in the trajectory. This type of segmentation is not suitable for our method since small changes in the trajectory should not result in dividing it, and we are only interested in removing the parts where there is no activity.

Another method that was proposed for trajectory segmentation is STC-SMo [23]. STC-SMo was initially proposed for the stop and move segmentation of human trajectories. The way it works is to find points that within their neighborhood, the duration of the longest time between points is greater than a threshold. It then labels these points and goes to the next point for the comparison. Although it can give satisfactory results for detecting stay points, this method cannot be adapted for the activity segmentation due to the diverse duration of different activities.

DB-SMoT is a segmentation method that is based on direction change in the trajectory data [24]. This method calculates the direction change for each pair of points and clusters the points based on that change. Each cluster starts with a sudden change and continues until the direction change continues. Another direction based method is [25], which uses direction for its first segmentation step. It calculates the angle between every two consecutive points with the first two points of the trajectory, and based on maximum and minimum

changes in direction, decides to divide the trajectory at that point or not. This step is their position fusion step. They have another step for sub-trajectory fusion that separates move and stops segments in them. Fishing vessels have many changes in their direction while fishing and if divided based on it, the sub-trajectories will not be meaningful.

Piecewise linear segmentation is a segmentation method for dividing a trajectory to smaller parts in a recursive way [26]. It starts with start and end points of a trajectory and calculates an error for all the points between them. At any point, if this error gets higher than a threshold, the trajectory is divided. The same procedure is done for the sub-trajectories that have been produced. Also, they have used a generalized error function with a weight parameter for time, which determines the effect of the time difference between points.

## 2.2 Non-maritime Trajectory Classification Research Works

There have been many works on trajectory classification that are not related to marine life; however, based on the high correlation that different trajectory datasets have with each other, these works can be applied for maritime-related problems. In order to prepare the reader's mind for the possible approaches that can be taken toward this type of data, several of these works have been briefly explained in this section.

Two of the main approaches that have been used in several works in this area are extracting global and local features for the trajectory to improve classification accuracy. Zheng [27] defines several global features like heading change rate, and stop rate that can help the model to identify the type of transportation. They assign labels for the segments of human trajectory based on these global features they extract for these segments. While global parameters can be useful for classifying the type of transportation, they cannot capture the similarities of the patterns in the movements.

One of the methods that uses both global and local features for classifying the trajectories is introduced in [28]. After preparing the trajectory data, they calculate the global features. They use an algorithm to convert the trajectories to time series of point features. These time series are made of the point features like speed and acceleration; then, they are decomposed based on the deviation to extract local features.

Change in the direction has been used in Lee [29] for extracting sub-trajectories from a trajectory. This work tries to group the sub-trajectories based on their movements. These

sub-trajectories can be in several groups based on their similarity, but they wanted a set of groups that have unique sub-trajectories. Different groups are made after this step, which some need to be removed only to keep the relevant ones. The relevance of groups is estimated based on the similarity of the groups. Therefore, they split the sub-trajectories again until the condition is met. The same procedure for classifying trajectories with a difference in using duration of the sub-trajectories has been used in [21]. They consider time and location to calculate the duration for different sub-trajectories to group them. Both of these methods make a feature vector for the trajectory instances based on how many of their sub-trajectories are in different groups. Classification is done based on the vectors that are extracted for the instances.

Movelets is another method that uses local features for sub-trajectories [30]. They do not have predefined criteria or threshold; instead they use the shapelets concept [31], for extracting the sub-trajectories and calculate their local features for classifying them.

There have been several research works around using Neural Networks for trajectory data classification [32, 33, 34]. One of the first works in this area is [32]. In their work, they have used features like horizontal dilution of precision (HDOP) of the data for each sample to train their network with. Although their work can be useful for having a guideline to use NNs in this area, based on their choice of extracted features, the results were not satisfactory. The performance of NNs highly depends on the extracted features of the data, and these types of features are not always available for trajectory datasets. In another work, the features that have been used are general features like the average and maximum speed that can be calculated from the raw point-based data [33]. Although their work improved the previous model's detection performance, the simple features alongside the simple architecture they used, are not sufficient for the task. More sophisticated works that use state-of-the-art models [34], are developed for trajectory related tasks. They have used Semi-supervised Generative Adversarial Networks (GANs) with fixed size segments of the trajectories.

A high number of methods have been focused on extracting more and more complicated features from the data points in order to improve the performance. For example, features like mean velocity, expectation of velocity, top three velocities, and the top three accelerations have been used in some works for trajectory classification [35]. Furthermore, the same author has developed more complicated features like the heading change rate (HCR)

and the stop rate (SR) for better understanding the trajectory of animals [27]. Though these features can be helpful in some tasks, in many cases like fishing, these features do not capture the pattern and cannot give enough knowledge for the classifier to predict the label. On the other hand, another set of methods have used additional data in combination with trajectory data to increase the accuracy of their method [36, 37, 38, 39]. Datasets that have been used range from sensor data for recording human body temperature to distinguish between running and walking, to bus GPS data for transportation classification.

## 2.3 Maritime Trajectory Classification Research Works

The dataset that we have worked on, as explained later, is a trajectory dataset that monitors the location of different types of vessels in the ocean. In this section, research works that have been explicitly trying to work on marine-related trajectory problems are summarized.

One of the main approaches that was first used for detecting fishing activities is the use of speed [40]. Based on the changes and frequency of the changes in speed, different fishing segmentation methods for some specific gear types have been proposed. Authors of [10], have tried to develop different models based on the vessel type. Their dataset is labeled manually, and therefore, it is limited. The vessel types they try to develop a model for are trawlers, longliners, and purse seiners. The first model they developed is a Hidden Markow Model on the speed of the trajectories which was trained for detecting the fishing activity of trawlers. For longliners, a pattern recognition method, and for purse seiners a multi-layer filtering of speed and time has been used. Despite their great accuracy for all the models, their data was highly preprocessd, so their methods are not capable of detecting fishing activities from noisy unlabeled real-world datasets.

Several supervised learning methods have been introduced to solve the real-world problems associated with trajectory data. SVMs have been widely used for marine trajectory classification. They make boundaries between clusters and map the unseen inputs to those clusters based on the calculated features. Trajectory kernels for SVM have been used in [41], which can improve the performance. However, their data was not big enough for the right comparison.

Artificial Neural Networks (ANNs) have been used for marine trajectory data mining related tasks too. In [42, 43] ANNs have been used with the Vessel Monitoring System (VMS) data to detect fishing activities.

Decision Trees (DTs) can be easily understood and implemented as a result of their simple procedure of hierarchy condition structure for classification. DTs are not used alone for trajectory classification because they heavily depend on the changes in the values, which frequently occurs in trajectory datasets. However, they are combined with other methods to remove useless features from the AIS data to distinguish between vessels [44]. Random Forest is a type of DT that takes advantage of bootstrapping idea. RFs have been used in [45, 46], for fishing activity detection and fishing gear classification.

A few marine trajectory clustering algorithms have been proposed that could give useful information about different classes and simplify the process of classification. In [47], a distance feature based on the Frechet distance for capturing the similarity between trajectories is calculated. Then by using Principal Component Analysis(PCA) on these distances, they decide on the number of clusters. Finally, based on this similarity matrix and the number of clusters, trajectories are clustered. Li et al. [48] Proposed a multi-step clustering algorithm for ship trajectories. They first use Dynamic Time Warping (DTW) for calculating the similarity between the trajectories, and by applying PCA, just like the previous work, the number of clusters by an automatic algorithm is chosen. Then the cluster analysis based on the similarities of the trajectories is done. Their experiments show satisfactory results that can give prior knowledge of data before the classification task.

Although deep approaches that have been introduced for maritime trajectory classification tasks have satisfactory results, they significantly differ from our method in the preprocessing steps and used architectures. The problem of fishing classification using Autoencoders was proposed by Jiang [45]. In this work, a similar process to our method has been used for extracting matrices of a sliding window. In their work, instead of using images as input small extracted matrices for sub-trajectories have been used as the input. Also, instead of a CNN, an Autoencoder has been used, and the global and local features of the trajectories have not been used for matrices. Also, segmentation is done by a sliding window of time. It is also worth mentioning that Autoencoders try to capture as much information as possible, and this information is not necessarily the most useful one for classification.

In another work, Jiang uses Recurrent Neural Networks(RNN) on partition-wise features of the trajectory data, [49]. That work tries to divide each feature into distinct regions and gives new features based on them. This process is done to use low-dimensional data

like trajectory with an RNN that can work best with high-dimensional data. Also, different architectures have been tested in their work that can reach satisfying performances; however, their method needs labeled data, and the amount of labeled data that is available is limited.

All of the above methods are supervised and need manually labeled data. However, labeled trajectory datasets are relatively small and limited, and the methods that have been trained on them can not be used for the tasks that the big amount of data enables us to solve. Therefore, only our method satisfies the need for an unsupervised method that can detect fishing activities. Our method combines the deep learning methods with DBSCAN segmentation method on the point-based representation of trajectories, to form the unique approach of detecting fishing activities from an image representation of the trajectories.

# Chapter 3

# Preliminaries

In this thesis, we have used different methods for each step, of which some need more explanation so that the reader can understand the method easier. This chapter defines and summarizes some of the terms used during the curse of this thesis.

## 3.1 DBSCAN

Cluster analysis tries to map the data points into groups or objects that are similar. In other words, a cluster is a group of points that have the most similarity with other points within the cluster and are different from points of the other clusters. Cluster analysis result is sometimes used as a preprocessing step for further classification and other tasks like our method, while other times, it is used for for giving useful insight about the data.

DBSCAN is a cluster analysis algorithm that was introduced in [50]. This algorithm separates the data points to different clusters based on the density in the neighborhood of the points. DBSCAN is robust, hence noisy data can be well clustered using it. Time complexity of DBSCAN is $\mathcal{O}(n^2)$, so it is not suitable for real-time clustering of big datasets. DBSCAN clusters the data based on the number of neighbor points each data point has. The main procedure is to look to form a new cluster when it finds a new dense region in the data. All of the following definitions are from [50].

MinPts and Eps are parameters of DBSCAN, and based on them, the data points are categorized, and clusters are formed consequently. The parameters are explained below[50]:

- MinPts: Minimum number of neighbor points that a core point should have.
- Eps: The points' neighborhood radius size.

$$N(p) = \forall q \in D | p \neq q \wedge dist(p, q) \leq Eps \tag{3.1}$$

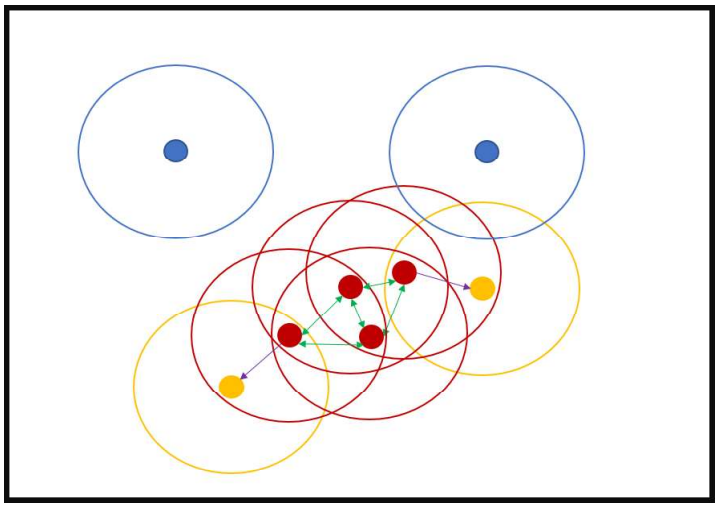


Figure 3.1: An example of categorizing data points in a dataset by DBSCAN. Red for core points, yellow for border points, and blue for noise points have been used. The two yellow border points are density reachable, and the arrows show directly density reachable relations [2].

The first step is to choose the parameters for DBSCAN based on the information you have about the data, which requires trial and error procedure. Next, based on the number of the samples in the neighborhood of each point (3.1), it tries to categorize the points as one of the core point, border point, or noise point. In this equation, D is our dataset, and N(p) is the number of points in the Eps neighborhood of point p. Different categories are explained as follows[50]:

- Core point: Any point with MinPts or a higher number of points in its neighborhood.
- Border point: All the points that are not core points but are in the Eps neighborhood of a core point are border points.
- Noise point: Any point that does not belong to the previous categories.

Different relations that points can have are[50]:

- Directly Density Reachable: If a point is within a core point's neighborhood, it is directly density reachable from that core point.
- Density Reachable: Two points are density reachable if there are core points between them to make a chain of directly density reachable core points that connect these two points.

- Density Connected: Two points that both of them are density reachable from another point, are density connected.

The algorithm of DBSCAN is to choose a random point and count its neighbor points. If that point is a core point a new cluster is formed, and that point and all of its neighbors are added to that clusters. The same procedure will be done for the points in the new cluster, so more points are added to the cluster in this process. When there is no more point to be added to the cluster, it is formed, and another random unvisited point will be visited to form more clusters. This recursive procedure is done until all of the points are visited. Any point that does not belong to any cluster is considered noise.

In Figure 3.1, an example of how points are categorized in the processes of forming clusters can be seen.

## 3.2 Linear Regression

Regression is a method that tries to find a relation between several variables, of which one is dependent on the other independent variables. Regression is primarily used for forecasting or predicting the dependent variable; Therefore, it is highly used in the machine learning field [51].

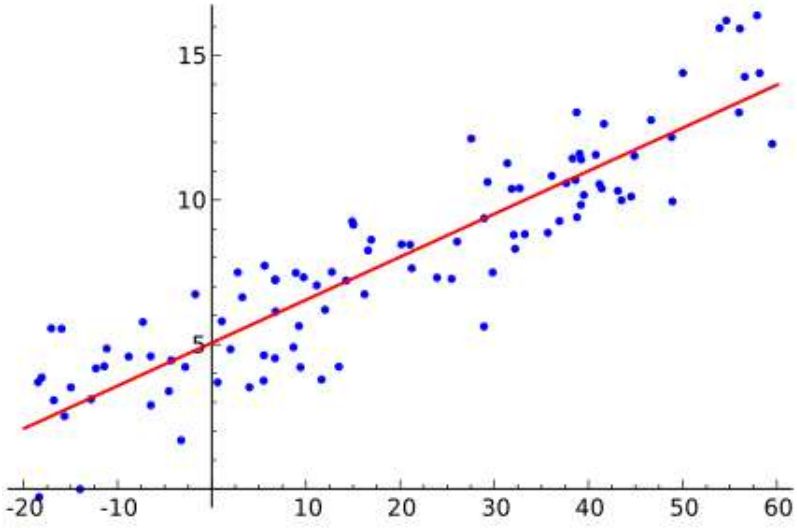



Figure 3.2: An example of finding the relation between variables with Linear Regression is shown. The red line is the predicted linear function that fits the best to the data points [3].

Linear regression is a simple regression method that tries to model the relation between

the variables as a linear function [52]. For two variables x and y, with y being the dependent variable, linear regression tries to find $\alpha$ and $\beta$ parameters which best satisfy the following equation:

$$y_i = \alpha * x_i + \beta i = 1, ...., n \tag{3.2}$$

In Figure 3.2, you can see that the linear model has been shown by a red line, which is achieved by finding the parameters from the data.

For understanding how well the model can fit the data, R-squared is used. R-squared is a measure that shows how well the predicted linear model is and it is mostly used for linear regression models. Higher values of R-squared show smaller differences between actual data points and the predicted line.

## 3.3 Deep Learning

Deep learning is the name given to a group of machine learning methods that are based on Artificial Neural Networks and have a high number of layers for learning more features from the data [53]. ANNs have been studied for many years [54, 55, 56, 57, 58], and deep learning methods became more famous from those researches.

Neural Networks are made of artificial neurons that are inspired by the actual neurons in a biological brain. There are connections between these neurons that based on the received values, can be activated and sent the values to other neurons, Figure 3.3. These values are real numbers and inside each neuron, some non-linear function produces an output. Connections between the neurons have weights that are learned during the process of training. In the process of learning, back propagation is used. Back propagating is a gradient descent method to slowly change the value of weights based on the label of the inputs to find the optimal values for them.

### 3.3.1 Convolutional Neural Network

Deep learning architectures are used in many fields, like Computer Vision, NLP, and Speech Recognition. Convolutional Neural Networks are one of the sub-types of deep learning methods that have great performance over two-dimensional data like images that have spatial correlation [59].

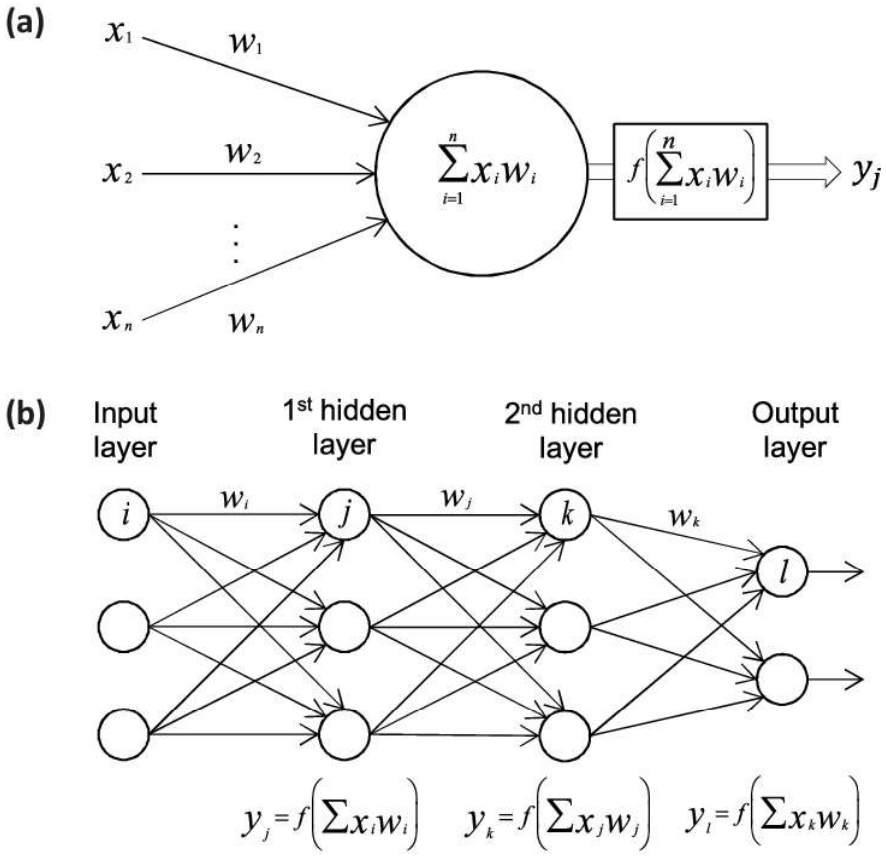



Figure 3.3: (a) Shows a neuron and its non-linear activation function that sends the produced output from applying that function to the weighted sum of input x, to other neurons. (b) Shows an example of a simple Neural Network that has two hidden layers. This figure is taken from [4].

CNN is a multi-layer neural network that has convolutional and pooling layers [60, 61, 62]. One of the main differences between CNNs and NNs is the use of convolution operation for each node instead of the weighted sum that we had in NNs. In addition, CNNs take advantage of parameter sharing by applying convolutional layers with several kernels on all the receptive fields of the previous layer. The input of the network is connected to a series of consecutive convolutional and pooling layers that try to extract features from it, and finally, the output features are connected to the output layer, Figure 3.4. The input of the networks is convolved with a series of kernels that are learned during the training phase, and the extracted features in the form of feature maps are produced. Because of an increase in the size of the feature map in comparison to the input, pooling layers are applied to reduce the size and get rid of the unimportant features [60, 61, 62].

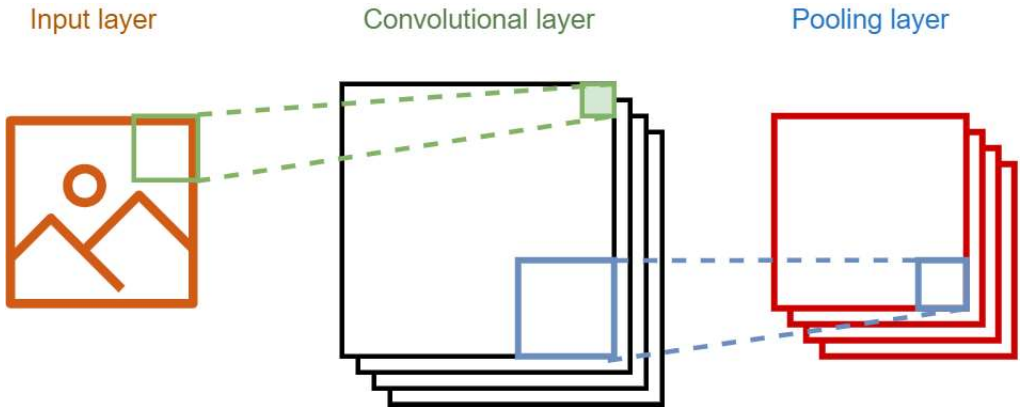



Figure 3.4: A simple example of applying convolutional and pooling layers on an input image, that shows how the size and number of values changes.

### 3.4 Trajectory Data

In this section, trajectory data is explained in more detail. Trajectory data, as defined in [1], is a set of ordered points in time that contain the location of a moving object. These points make a trace that is called a **spatial trajectory**. For example a set of consecutive points like $x_1, x_2, ..., x_n$ Where each point $x_i$ represents a point that contains location and time, is a trajectory [1]. Figure 3.5 shows an example of a spatial trajectory.

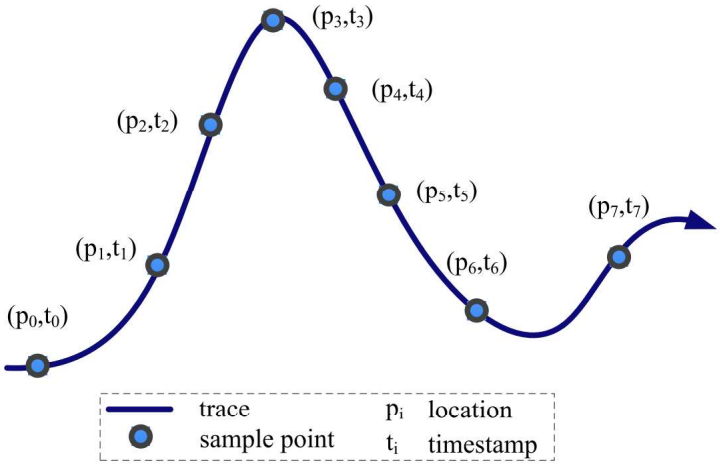
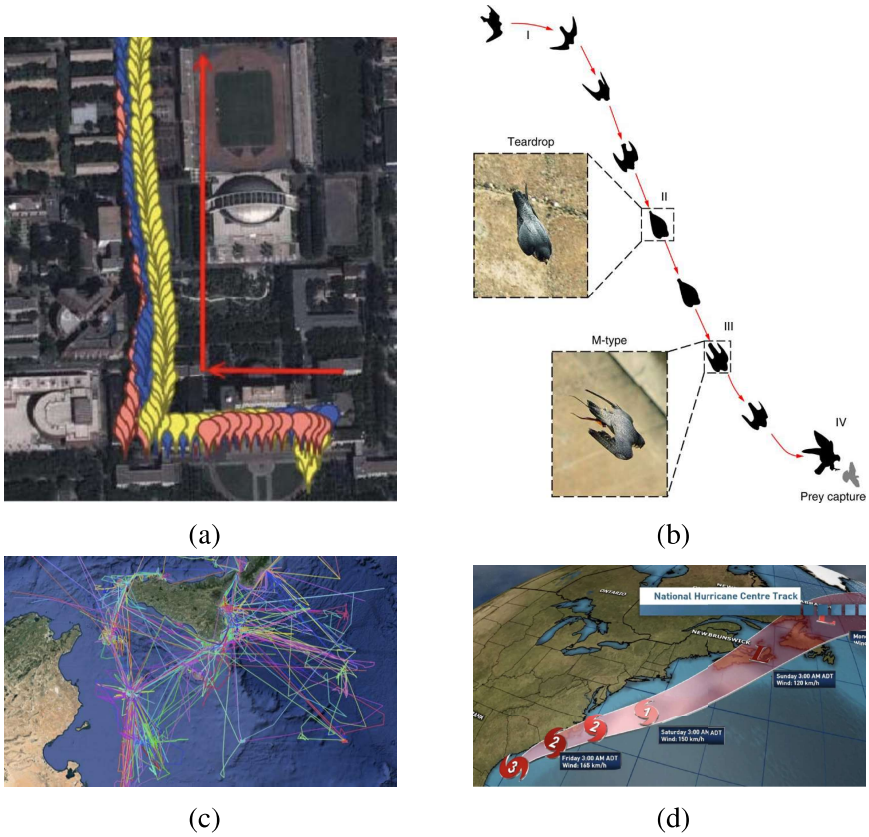



Figure 3.5: An example of a spatial trajectory made of consecutive points. Source: [5].

Four main categories of trajectory data are [1]:

- Mobility of people: Movement of humans has been gathered through different means.




(a) (b)

(c) (d)

Figure 3.6: Examples of different categories of trajectoriy data:(a) Mobility of people [6], (b) Mobility of birds [7], (c) Mobility of ships [8], (d) Mobility of hurricane [9].

Some people intentionally have recorded their location at different points while traveling or other activities. On the other hand, GPS can record the location of people holding a cell phone through cell towers without them knowing about it. Many trajectory datasets have been collected from cell phone users.

- Mobility of transportation vehicles: different types of transportation vehicles on land, air, and water are equipped with GPS tools, so large trajectory datasets of their movement are recorded.

- Mobility of animals: Movement of animals for biological purposes have been recorded during time. These movements are saved in trajectory datasets for animals that can be analyzed.

- Mobility of natural phenomena: With the rise of concerns about climate change, many scientists have gathered trajectory data for natural phenomena like hurricanes

to capture their changes.

Our data is trajectory data of vessels on the ocean, which is categorized as the Mobility of transportation vehicles. Examples of different categories of trajectory data are shown in Figure 3.6.

### 3.4.1 Gear Type of Fishing Vessels

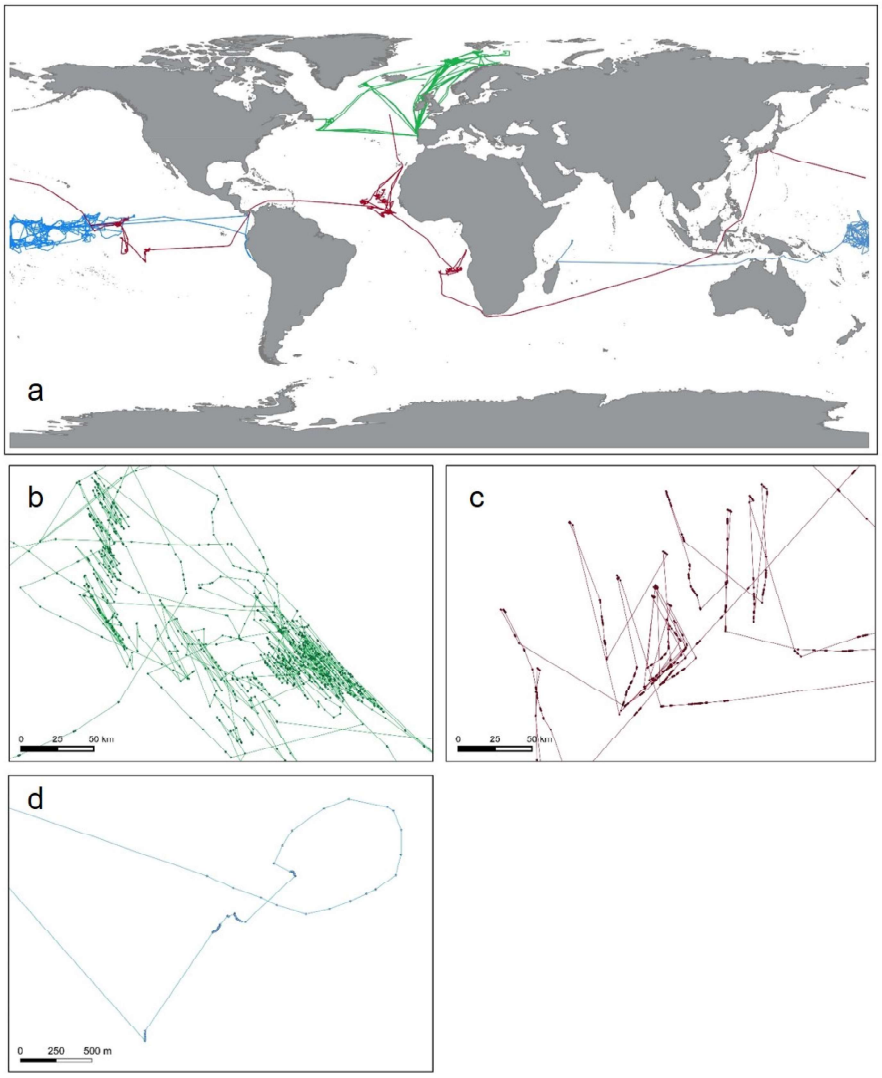



Figure 3.7: Plotted raw trajectory data on (a) a global map, of (b) Trawler, (c) Longliner, and (d) Purse Seiner gear type vessels. These plotted trajectories show the difference in patterns of fishing activities for different gear types of fishing vessels. This Figure has been taken from [10].

There are different types of fishing vessels that each one of them has its pattern and

characteristic for fishing. Fishing detection methods can be dependent or independent on the gear type of fishing vessels. Some of the methods that have been mentioned in the Related Work chapter, have different models for each type of fishing gear. The most common fishing gear types, Trawler, Longliner, and Purse seiner, are explained in this section.

### Trawler

Trawler is a type of fishing vessel that attempts to catch fish through dragging and pulling a fishing net behind the vessel. This gear type has different categories itself, including but not limited to, Freezer trawlers, Side trawlers, and Pair trawlers. These types may differ in the number of vessels that fish together, the depth of fishing, or the type of fishing nets they use. Duration of the fishing activity for the trawlers highly depends on its category and can vary from minutes to hours; however, the speed of trawlers is slow and steady, almost for all of the categories [63].

### Longliner

Longliner is a type of fishing vessel that has a long line attached to the back of it. This line can have hundreds to thousands of hooks attached to it so that when the vessel is moving, and the line is deployed at the back, the fish will be caught by those hooks [64]. The duration of this gear type activity depends on the length of the line and can be as long as a day [65]. Also, the speed is slightly slower than the speed of the vessel when it normally moves without doing any activity. Longliners have different categories, and the length of the line and number of the hooks depends on it, which consequently can change the speed and duration of fishing [65].

### Purse Seiner

Purse seiner's fishing activity is the process of deploying a vertical long hanging net around an area to prevent fish from escaping from that area. In order to avoid the fish from escaping, this process should be fast; therefore, the speed of purse seiners while fishing is usually high. The duration of this process is based on the size of the area and net. so it varies from one to a couple of hours [10].

You can see in Figure 3.7, each of these gear types have their specific patterns that can help the experts to label the trajectories. As a result, we came up with the idea of

converting the task of fishing detection to an image classification task that would be general and independent of the gear type. Also, speed is one of the main differences between different gear types and gives useful insights about each pattern; therefore, our method also has used speed for creating the trajectory images.

# Chapter 4

# Proposed Method

In this chapter, the proposed method is explained step by step in full detail. From preprocessing the raw trajectories to creating images and training a classifier on them, our method is made of the following main steps:

(i) Extracting AIS trajectory data and separating it into fishing versus non-fishing trajectories automatically based on ship type code.

(ii) Prepossessing and segmenting spatial trajectories.

(iii) Converting sub-trajectories to images.

(iv) Training and testing a CNN on the created images.

The flowchart presenting how these steps are connected is shown in Figure 4.1. The first step is to extract the trajectories from the regions in which fishing rates are high. Then preprocessing and segmentation steps are done for each region, and images are created. Next, when the image dataset is complete, a CNN is trained and tested using that dataset, and finally, the results are shown.

## 4.1 AIS Dataset

The data that we have used is mainly AIS data that has been extracted for the regions containing high fishing activity. The exact location of the regions and the time periods of the extracted data will be explained later in Experiments Chapter.

AIS is a system for monitoring the ships on the ocean. AIS records the location of the vessels using transponders, which are mounted on the ships. The data is used to avoid unwanted incidents on the ocean [66]. AIS uses a combination of several types of receivers and sensors like GPS and very high frequency transceiver, to record the exact location of the ships along with some other features like speed. This data is usually used by the vessels to monitor the traffic in their area, and allow other vessels to know about their presence.

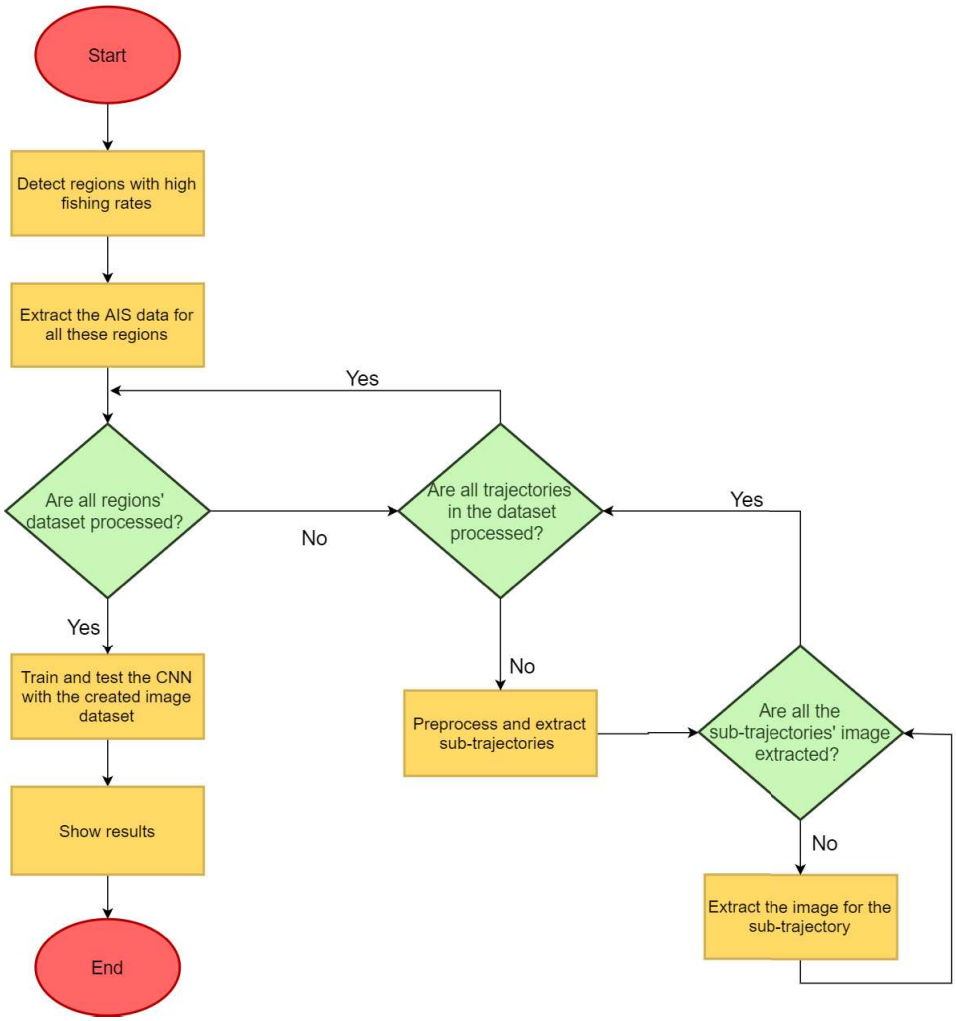



Figure 4.1: Flow chart of the proposed method.

Satellite-AIS (S-AIS) is the term used when satellites are used to record the AIS data where the vessels are out of the range. There have been several different methods to address the issue of receiving large amounts of AIS messages, for different classes of AIS, simultaneously through satellites. These methods help to extend the ranges that AIS could cover [66]. Our data has been extracted from both traditional AIS and S-AIS sources.

Another essential concept to be mentioned about AIS data is different classes of AIS data, which differ in the type of device they use for communication. Type of the device and consequently, the AIS class of a vessel, is determined based on the mandate of it [66]. Out of all the classes, class A and class B have been briefly explained here, as our data only includes messages from these two classes.

**Class A:**

This class requires devices that meet the International Maritime Organization (IMO) carriage requirements, and these devices are found on board vessels [66, 67]. These devices report the position every 2-10 seconds while their statistic information is reported every 6 minutes. Class A is also capable of communicating through text messages for safety-related situations [67].

**Class B:**

Class B devices are mainly found on vessels that voluntarily carry AIS; they are generally less expensive, and they do not meet all IMO requirements. Although they report their position at anchor or moored as often as Class A, less power is used for reporting their position. In addition, they are not able to transmit text messages and can only receive them [66, 67].

The data containing position and statistic information of both classes have some common fields which are used in our method. These fields are explained for both position reports and static and voyage related data.

Common fields of position report for both classes are [11]:

- **Message ID:** There are different message types for different classes. This ID tells what type of message has been used.
- **User ID:** MMSI number which is a unique identification number for ships.
- **SOG:** Speed over ground is used for each point.
- **Time stamp:** The time when the message was generated by the device.
- **Longitude:** Longitude of the vessel when the message was generated.
- **Latitude:** Latitude of the vessel when the message was generated.

The only field of static data that has been used is the type of ship and cargo type. This identifier helps us to distinguish fishing vessels from non-fishing vessels. The vessels should report their type based on the table shown in Figure 4.2, [11].

| Identifiers to be used by ships to report their type | | | |
|---|---|---|---|
| **Other ships** | | | |
| **First digit(1)** | **Second digit(1)** | **First digit(1)** | **Second digit(1)** |
| 1 - Reserved for future use | 0 - All ships of this type | - | 0 - Fishing |
| 2 - WIG | 1 - Carrying DG, HS, or MP, IMO hazard or pollutant category X | - | 1 - Towing |
| 3 - See right column | 2 - Carrying DG, HS, or MP, IMO hazard or pollutant category Y | 3 - Vessel | 2 - Towing and length of the tow exceeds 200 m or breadth exceeds 25 m |
| 4 - HSC | 3 - Carrying DG, HS, or MP, IMO hazard or pollutant category Z | - | 3 - Engaged in dredging or underwater operations |
| 5 - See above | 4 - Carrying DG, HS, or MP, IMO hazard or pollutant category OS | - | 4 - Engaged in diving operations |
| | 5 - Reserved for future use | - | 5 - Engaged in military operations |
| 6 - Passenger ships | 6 - Reserved for future use | - | 6 - Sailing |
| 7 - Cargo ships | 7 - Reserved for future use | - | 7 - Pleasure craft |
| 8 - Tanker(s) | 8 - Reserved for future use | - | 8 - Reserved for future use |
| 9 - Other types of ship | 9 - No additional information | - | 9 - Reserved for future use |

Figure 4.2: This table from [11] shows how to interpret the ship type code to distinguish between fishing vs non-fishing vessels.

As you can see in Figure 4.2, the code for fishing vessels is **30**. Therefore after downloading the data, one of the first steps is to separate the trajectories that have a ship type code **30** as our fishing data and all the other samples are labeled as non-fishing data.

Furthermore, before moving further to the segmentation part, the noise and outliers from the data such as points that have an MMSI code of **0**, or other non-values, are removed. As we are only interested in sub-trajectories that contain fishing activity for the fishing data, the points that have a SOG equal to **0** are removed from the data.

## 4.2 Segmentation

After the separation of fishing versus non-fishing trajectories and other preprocessing steps of the data, we extract sub-trajectories. These sub-trajectories are made of important parts of the bigger trajectories where the vessel has been doing any type of activity; for this purpose, segmentation methods have been used.

A segment is defined as a part of a trajectory in which the vessel's movement has a particular pattern. For example, a fishing vessel starts moving toward a destination for fishing; then, for hours, the vessel is engaged in the activity. When the fishing is done, the vessel comes back to where it left. In the given example, a segment consists of the points where the vessel starts fishing until the process is completed. The recorded points of the vessel on the way and back are not of interest in our method. therefore, a contiguous part of a trajectory so that all points in it share the same pattern.

The first step for the segmentation part is to detect the points that belong to the same trajectory and group them together. For this purpose, we assign an id to the points that have the same MMSI, which indicates the trajectory number of that particular vessel. For example, a vessel with MMSI number **m** has recorded his movement in three different days during a year; therefore, three different trajectories with the same MMSI number exist that we need to separate. This step is needed to avoid having trajectories with the same MMSI but at different times to be grouped together. Time is not considered for clustering the points, so this step is essential.

The next step is to take a trajectory and convert the latitude and longitude of all its points to radian. Radian is a unit for measuring angles and is used in many problems in which the data consists of latitudes and longitudes. The range of the data will be in $[0, 2\pi]$ after the conversion. DBSCAN's performance on the longitude and latitude that are not converted to radian is not satisfactory. In addition, as we define our Eps parameter of DBSCAN in radian, this step is necessary. After having a suitable scale for our input, the points of a single trajectory are fed into the DBSCAN. Parts of the trajectories, if any, where that specific vessel has been doing an activity, are returned as clusters.

As mentioned earlier, there are several other segmentation methods [23, 25, 22, 23, 24], but they are not suitable for our segmentation steps because of several reasons. The first reason is that many of these algorithms have very high time complexities. As we need to run our segmentation method on every single trajectory for a huge dataset, the time and memory they need are not reasonable.

The next reason is that these methods usually try to extract sub-trajectories that are considerably smaller than what we need. For example, the methods that separate a sub-trajectory based on the rotation would break a sub-trajectory, made of fishing activity of a longliner to several smaller parts which are not useful for learning the pattern of longliner fishing. As it can be seen in Figure 4.3, the pattern of a longliner's fishing activity has been broken into several single lines if separated by rotation.

Some segmentation methods that are presented for stay point detection are not useful in our case, as they usually limit the radius of the sub-trajectory. On the other hand, fishing vessels have different patterns and can travel small to large areas while fishing. As a result, these types of segmentation methods can not be used for our model.

Finally, some methods that try to break a trajectory into sub-trajectories focus on the

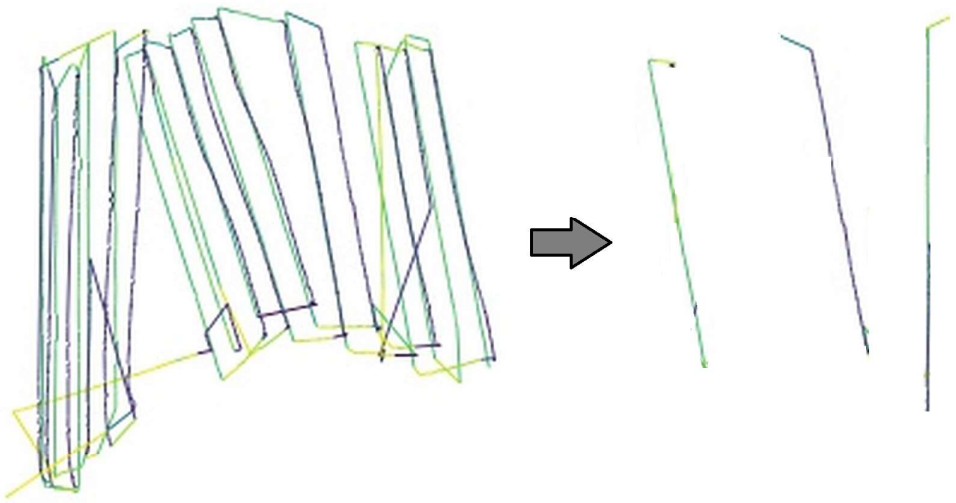


Figure 4.3: The sub-trajectory that shows the fishing activity of a vessel, will break into smaller sub-trajectories if an inadequate method is used.

change of speed. The problem that was mentioned above for segmenting with rotation will also happen when segmentation is done using speed; because speed of vessels is subject to change during fishing activity based on the patterns. As a result of the mentioned problems, we have used DBSCAN for our segmentation step.

DBSCAN is applied on the latitude and longitude of each point in the trajectory, and we do not consider any other feature of the data for this. Therefore, the input of DBSCAN is a set of two dimensional data points. After getting the results of DBSCAN for each trajectory, for all the output sub-trajectories that DBSCAN gives as clusters, several conditions and filters are applied. If a sub-trajectory meets all the requirements, it will be given to the image creation algorithm to convert the sub-trajectory to an image. The filters and conditions that need to be met are:

**Time Limit**

The duration of fishing activity, as well as other activities that we want to create the image of, can vary and is not stable. However, we are not interested in sub-trajectories that are as short as 2 hours, as they are a result of sudden sparse regions in the middle of activities that DBSCAN will break into smaller clusters. This type of sub-trajectories that have been separated by mistake using DBSCAN, are removed using a time limit for the duration of an activity.

### Average Speed Limit

One of the cases that can be easily mistaken for an activity sub-trajectory using DBSCAN is where the vessel is not moving. Stay points can easily make a cluster, so some patterns can be extracted if the lines between these points are drawn. Therefore we use another threshold for the mean speed in each sub-trajectory, and if it is close to 0, the sub-trajectory is removed.

### Number of Points

Another factor when considering a sub-trajectory as an activity is the number of its points. A small number of points in a sub-trajectory can be a result of single small regions with high density that cannot show the whole activity as a sub-trajectory. Therefore these small dense regions that do not contain enough points for pattern learning are removed with a threshold for the minimum number of points in a sub-trajectory.

### Removing Dense Non-activity Sub-trajectories

Use of DBSCAN for our application has some limitations because of the global parameters it uses. Some points in the data that have been recorded while a vessel is moving toward a destination can be detected as a cluster with DBSCAN. The density around these points can satisfy the DBSCAN parameters if the vessel is transmitting its location more frequent. Therefore, some of the clusters that DBSCAN returns need to be removed as they do not capture any activity pattern. A few examples of these clusters are shown in Figure 4.4.

We use Linear Regression to identify the non-informative sub-trajectories that need to be removed. Linear regression tries to map points of a sub-trajectory to a linear model. Then, the R-Squared of the model, which is a metric for the performance of the model is calculated. A sub-trajectory is removed if the model can fit the points of the sub-trajectory well. In other words, if the R-Squared is higher than a threshold, the sub-trajectory is not informative and will be removed. Several examples of where the linear regression model can remove the dense non-activity sub-trajectories that are noise samples for our CNN model is shown in Figure 4.4.

Once all of these conditions are met, and the sub-trajectory is not removed, image creation method will draw it and save it as an image in the new dataset.

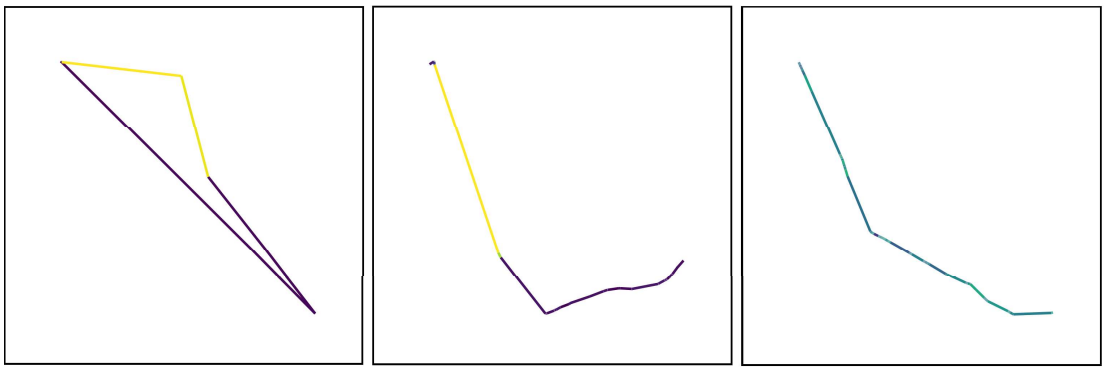


Figure 4.4: All of the above examples of sub-trajectories that are produced by DBSCAN are removed using linear regression filter.

## 4.3 Image Creation

Image creation is the part where sub-trajectories which have passed all the filter, are converted to an image. The process of drawing the sub-trajectories and saving them as images is explained in detail.

For drawing the lines between the points, we have converted the points to the radian system and sorted them based on time. Then we start by the first point, and up to the last point in the sub-trajectory, the lines between them, which make the pattern, are drawn.

One of the critical features that can help the model to detect fishing activities, as discussed before, is speed. Different vessels with different patterns can be detected easier by considering the change of speed. For example, one of the methods that have been proposed to detect fishing activities of trawlers only uses speed as their feature to train a model with [10], which gains satisfactory accuracy.

As the importance of speed, we decided to make use of speed as the color of lines that connects the points. Therefore, the change in speed can be shown as a part of the pattern of fishing. The repetitive pattern in the speed can be seen in Figure 4.3, which can help the model to understand the patterns better. The color of the lines in that pattern shows the difference in the speed of the vessel in the parts where it is turning, in comparison to the parts it is moving in a straight line. As a results of using speed in the process of creating images, the created dataset contains colored images.

After creating all the images and forming our dataset, a final step of removing images based on size is done. The images that are made of sub-trajectories, despite all of the filters that have been put there to delete non-informative sub-trajectories, still contain a large

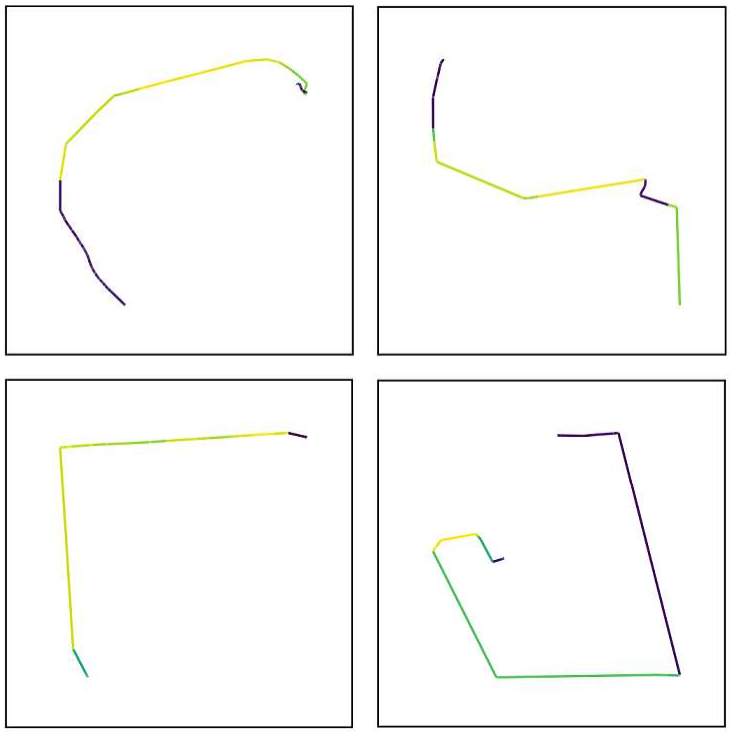


Figure 4.5: These images are created from non-informative sub-trajectories, which will be removed by our size filter.

number of useless patterns like the ones that are shown in Figure 4.5. These images are not detected by the linear regression, as they cannot be mapped into a simple line.

After deleting the non-informative images from the data, the new dataset is ready to be used for training our CNN models.

Our dataset is not labeled by experts, but we have used fishing versus non-fishing labels for our images. These labels are extracted using the ship type code. As explained earlier, the type of the ships in trajectory datasets is reported using a code. This code can be used for separating the fishing ships from others. The ship type code is simply used for evaluation and training purposes as label of images. The model can detect fishing activities form unseen trajectories without knowing this code. This code is not used as a feature of the input for the model, Figure 4.6.

We have assumed that the created images from the activity of the fishing ships in the middle of the ocean are fishing activities, as they do not perform other types of activities. For non-fishing data, any other image that is created from other ship types is considered non-fishing. The non-informative parts of fishing trajectories can be labeled as non-fishing

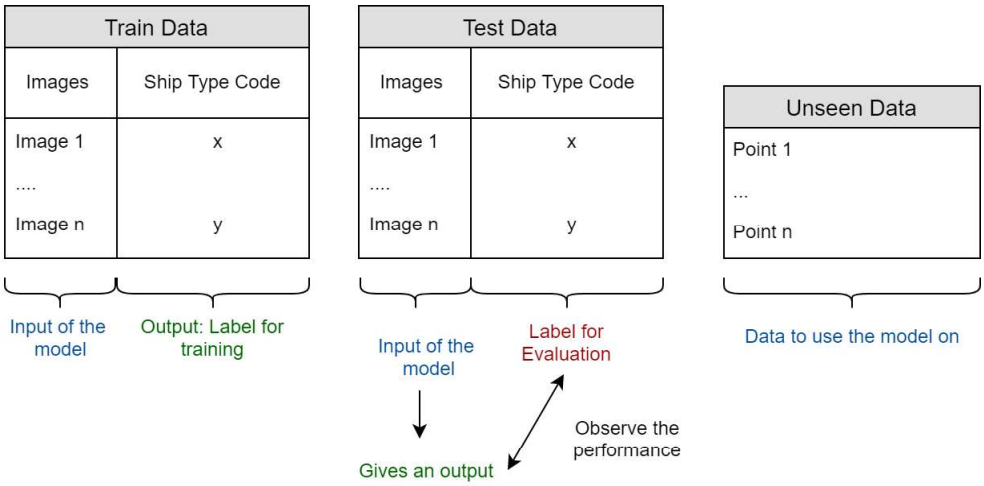



Figure 4.6: Usage of ship type code for our model.

data as well. However, as these parts do not contain any pattern and identifying them is easy, we decided to remove these parts.

## 4.4 CNNs for Trajectory Classification

The last part of our method is training a chosen CNN on our image dataset that has been created. Different CNN architectures have been proposed in the last decade, that each one of them has its characteristics. We have chosen ResNet [13] and Inception [12] for our case, and the results have been shown on both of these networks. In this chapter, we go over the networks that we have chosen and cover the detail about them. Finally, the reasons for choosing ResNet and Inception have been explained.

### 4.4.1 Inception

The first network that we have used is version three of the Inception Network. Inception is a network architecture that was first introduced in ”Going Deeper with Convolutions” [12]. There were several reasons for which this architecture was developed that improved the accuracy significantly. The main reasons were:

- Choosing the right kernel size is strenuous.
- Deeper networks can lead to overfitting the network.

- Adding more layers without rational reasoning makes the network computationally expensive.

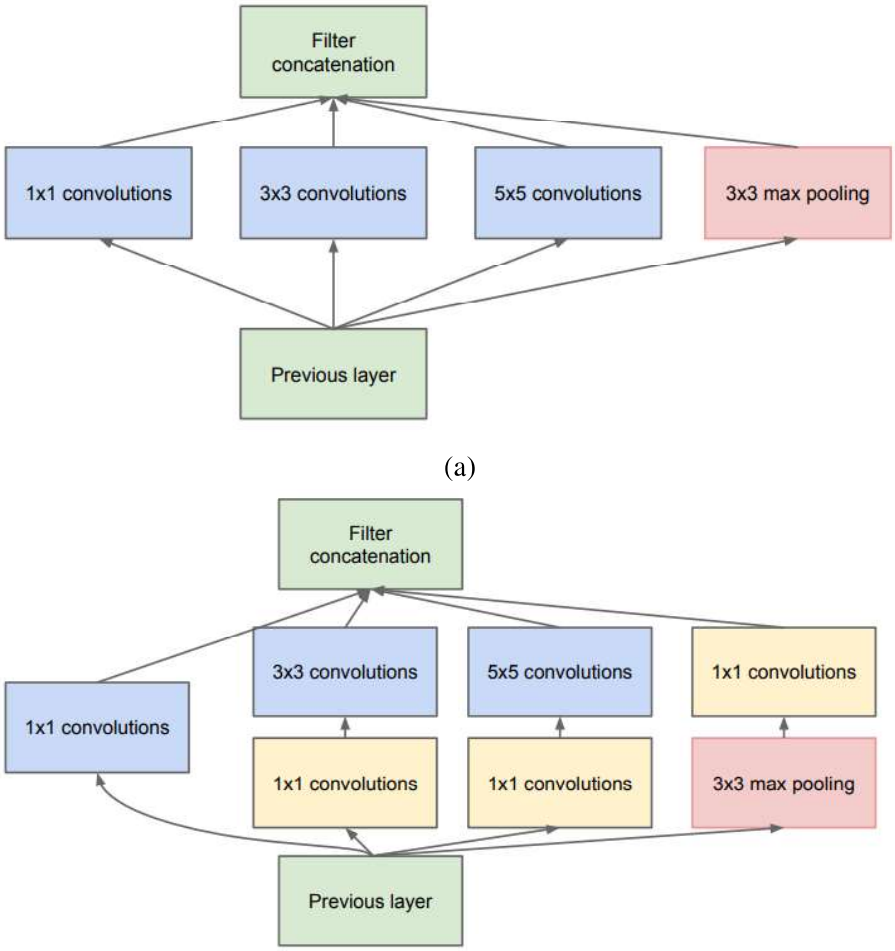



Figure 4.7: The basic idea of an Inception module. (b) is a version of (a) that has dimension reduction with the use of 1X1 convolutions. Source: [12].

With all the works focusing on creating deeper networks, different problems in the field arose. Deeper networks had different problems like vanishing gradient and overfitting, which Inception solved using the Inception module Figure 4.7.

The salient part of the images occupied by the patterns that needs to be classified have different sizes. This difference in the size of the useful part of the image arises the need for using different kernel sizes. As deciding on the right kernel size can be complicated and problem-dependent, Inception module was designed with different kernel sizes in each

module that leads to a wider network Figure 4.7(a). This module shows how the architecture takes advantage of several different kernel sizes in each layer.

How 1X1 kernels have been used for dimension reduction is shown in Figure 4.7(b). 1X1 kernels cannot capture any feature as they are applied on one pixel at a time. However, they are used to reduce the number of the input channels. As seen in Figure 4.7(b), they are applied before other kernels, so that the output of the module has a rational size. Finally, the outputs are concatenated and passed to the next layer.

In our task, the created images from the trajectories have different pattern sizes. Based on how the points of a segment for a specific activity are spread in the space and how they have been clustered, images will have different salient part sizes. For example, as it is depicted in Figure 3.7, different gear types have different salient part sizes in the images and subsequently different kernel sizes are needed. As a result, Inception has been used to satisfy this need of our dataset.

### 4.4.2 ResNet

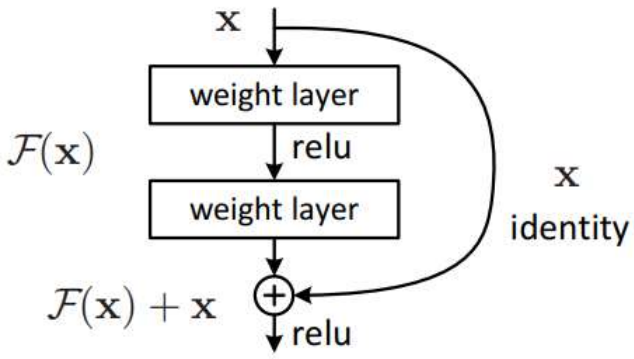



Figure 4.8: Main innovation of ResNet architecture was the use of shown residual block. Source: [13].

ResNet is the other architecture that we have used to train it with our dataset. A residual learning framework was first introduced in [13] to solve the issues that other deep neural networks had.

Researches showed that deeper networks had better performances on different tasks, while after a certain depth, the performance would start to deteriorate. Issues like generalization incapability and vanishing gradient were the main reasons for creating residual

blocks.

The main idea of a residual block, as shown in Figure 4.8, is to add the input to the output of the layer. Logically speaking, neural networks can easily approximate $f(x) = x$; therefore, adding the input to the output of a layer should not affect the capability of the networks to find $h(x) = g(x) + x$, which is the function it was learning plus the input [13].

The shortcut connection in the residual block enables the gradient to flow directly through them. As a result, the vanishing gradient problem can be solved, and deeper networks will have the desired performance. The residual block, including the shortcut connection, is used many times based on the number of layers that the network has.

As an experiment on how the depth of the CNN can affect the results of the method, ResNet architecture has been used for the fishing activity classification.

# Chapter 5

# Experiments

In this chapter, empirical results have been presented on the trajectory datasets from different areas of the world to show the performance of our presented method.

## 5.1 Dataset

As explained before, the dataset that we have used is a trajectory dataset that has been recorded using AIS and S-AIS technology. Our dataset contains different time periods from different regions, which is shown in more detail in Table 5.1.

The collected data was chosen from the specified regions based on the fishing activity rate. With the use of online tools and getting advice from experts, regions with high fishing activity rates were chosen, and the data of those regions were collected. On the other hand, there were no specific time periods of interest, and we added any available time period of those regions to our data.

In addition to all of the data that is extracted from various regions, another trajectory dataset, which was made specifically for a fishing detection competition, has been used to create the images of fishing activities [68]. The dataset is from a competition to detect the type of the fishing vessel for a given MMSI number. Many of the fields of this dataset are the same as the datasets that we have used. However, in our dataset, without considering the vessel type, we have simply used the competition trajectories as fishing activity samples

| **Class** | **Region** | **Time Period** | **Number of Vessels** |
|---|---|---|---|
| A | Barents | 2011-2019 | 4488 |
| A | Brazil | 2011-2019 | 23001 |
| A | Kiribati | 2011-2019 | 6305 |
| A | Galapagos | 2011-2019 | 9713 |
| B | Galapagos | 2011-2019 | 2532 |

Table 5.1: Regions and time periods of the extracted trajectory data that make our final dataset.

| Dataset | Total number of Images | Fishing Activity Images | Non-fishing Activity Images |
|---|---|---|---|
| A-Barents | 1911 | 757 | 1154 |
| A-Brazil | 7616 | 1317 | 6299 |
| A-Kiribati | 14258 | 4988 | 9270 |
| A-Galapagos | 2096 | 147 | 1949 |
| B-Galapagos | 1206 | 23 | 1183 |
| Competition Dataset | 1035 | 1035 | - |
| Total | 28122 | 8267 | 19855 |

Table 5.2: Number of the created images for each dataset we have, which are combined to make our final image dataset.

to create images from them. This dataset is relatively small, and only trajectories of 1025 vessels for short periods of time are included.

After all the preprocessing steps for creating images from the datasets that we had, a total of $28,122$ images were created, of which 29% were fishing activity images and the rest were non-fishing activity images. The exact number of images created for each of the smaller datasets that we have used is shown in Table 5.2. It is also worth mentioning that the size of the images is $224 \times 224$ pixels.

As seen from Table 5.2, the number of images that are created is lower than expected. This phenomenon is due to the difference in transmitting rates of ships. As explained previously, DBSCAN, with a set of global parameters, is used for extracting sub-trajectories that we apply filters on. These global parameters do not satisfy all different densities, so they can eliminate some of the useful sub-trajectories due to their lower densities. We could have set the parameters that are suitable for lower densities to create more images from the trajectories, but this would result in having a large amount of noise in our final image dataset.

Another problem with parameters suitable for lower densities is the connection of the activity part of the trajectory with the non-informative part of it. If the parameters satisfy the density around the path before and after the activity, the details of the activities will be unclear as a result of zoomed out images.

Setting the right Parameters for the DBSCAN can be a challenging task. We used trial and error to decide on the most suitable parameters for our case. These parameters can give an image dataset with a low number of non-informative segments, which can be omitted.

The parameters we have used are 8 and 80 for Eps and MinPts, respectively. With these parameters, an image dataset of the mentioned sizes in Table 5.2, has been created. Also it should be mentioned that in the trial and error process only a small part of the data has been used, and we have made sure that no test images have been created from that part.

All of the datasets have been combined to make our final trajectory image dataset. For training and evaluation purposes, we have split the data into train, validation, and test sets according to the $0.6 - 0.2 - 0.2$ proportions, respectively.

## 5.2 Results

In this section, the setting and parameters of the CNN models, along with the results, have been presented and analyzed. We will argue that our results show the significance usefulness of our model on real-world data.

The results of our method have been reported for Inception [12] and ResNet [13] models that we have used. However from different metrics, we have decided that Inception model has a better performance on this task and this model has been analyzed in more detail. We did not use cross-validation for our experiments because the available systems were not sufficient for it.

One of the parameters that is highly important in the training process is the Learning Rate (LR). LR is hyper-parameter that we set for our step size when updating our wights of the network [69]. Smaller LRs will lead to a need for a higher number of epochs, while bigger ones might result in undesirable performance. Therefore, we trained our models for only $10$ epochs with different LRs and based on results, the LR value of $0.003$ was used. Figure 5.1, shows the learning curve for our Inception model trained on the train data and evaluated on validation data for each epoch.

In terms of time complexity, our method can be used in real-time for predicting new samples. Our method has two main parts that need to be analyzed for the time complexity of it, image creation and training a CNN. The image creation part has a time complexity of $\mathcal{O}(n)$, with n being the number of trajectories. The process of creating images from trajectories, includes DBSCAN, applying filters, and drawing the pattern. All of these steps need a small time that is either constant or depends on the number of the points in a trajectory, which is limited. Therefore the time complexity of the image creation part is linear and can be done in real-time. The next part is training a CNN, which only will be

| Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Inception Image-based Model | 0.83 | 0.87 | 0.71 |
| ResNet Image-based Model | 0.82 | 0.87 | 0.68 |

Table 5.3: The results of our model on the created image dataset.

done once. Once the trained model is saved, it can be used for prediction. Therefore our model can predict fishing activities from unseen data as they are received.

Also, it is worth mentioning that the whole process of this work has been done in two parts on two different systems. The creation process of our image dataset on the Big Data institute's server with 40 CPUs of type Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, took a total of 6 hours and 10 minutes for all datasets. The learning process of the CNN Inception model has been done on Dalhousie DeepSense's systems. Two GPUs of type Mellanox SN 2700, have been used to train our model in 6 hours.

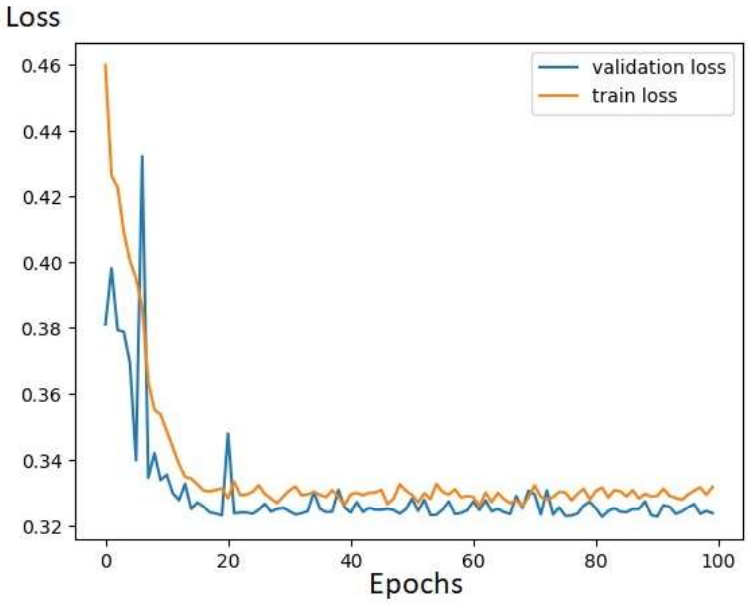



Figure 5.1: Learning Curve of the Inception model.

Thereafter the training process, we test our model on the test data. The performance of a model cannot be truly captured if a test set is not used, as the validation data has been used for parameter tuning. Our model on the test data has satisfactory results, which have been shown in Table 5.3.

| Model | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Souza [10](purse seiner) | 0.97 | 0.99 | 0.71 |
| Souza [10](longliner) | 0.83 | 0.57 | 0.93 |
| Souza [10](trawler) | 0.83 | 0.93 | 0.68 |

Table 5.4: The results of Souza's model on their expert labeled dataset for different fishing gear types.

The metrics that we have used are Accuracy, Sensitivity, and Specificity. The equations of these metrics are given in Equation 5.1. Sensitivity is related to non-fishing results, and specificity is related to the fishing detection results. The two later metrics are required to show that the model was not imbalanced, and results are interpreted correctly.

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + TN + FN} \\ Sensitivity &= \frac{TP}{TP + FN} \\ Specificity &= \frac{TN}{TN + FP} \end{aligned} \tag{5.1}$$

These parameters can show that the performance of our Inception model is satisfactory, Table 5.3. Some other models that have been trained for fishing detection of a specific gear type in [10] have slightly better results, Table 5.4. For example, Souza's method for fishing detection of purse seiners has excellent performance. However, these results are on a different small dataset that has been cleaned and labeled by experts. Therefore these higher numbers do not necessarily show better results for those methods in comparison to our method. The performance of our method is proved to be good enough on the noisy real-world data that has not been labeled manually.

We were not able to compare our method to Souza's method [10] on the same dataset. Our data is a big noisy dataset that has not been labeled by experts, and the gear type of fishing activities is not clear. Therefore Souza's method cannot be tested on our data. On the other hand, the data that they have used is a small dataset, and their results are calculated based on the number of points that are correctly classified. However, our method works with the number of correctly classified images, and as the dataset is small, this cannot be a useful metric for the experiment.

One of the main discussion in many research papers that train a CNN model is size of

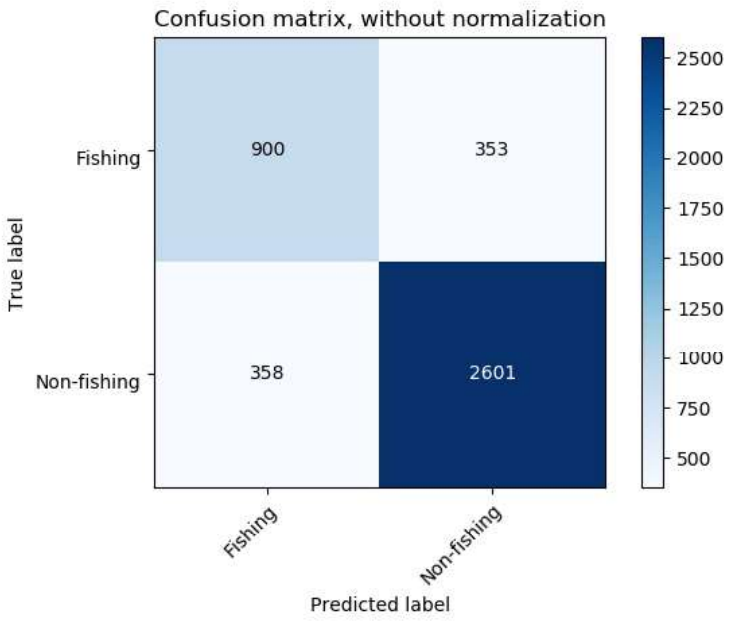



Figure 5.2: Confusion Matrix of the Inception model.

the kernels that are used. As explained in Chapter 4, the Inception model uses different kernel sizes in each layer of the network. Therefore, there is no need to manually try different kernel sizes because the model contains several of them.

In addition to the reported results with the defined metrics, the confusion matrix of our method has been shown in Figure 5.2. As can be seen from the number of False Negatives and False Positives, Our method has an excellent performance in terms of detecting fishing activities. There is a significant gap between the True and False predictions for our classification, which shows the strength of the method on unseen data.

We took a further step to look into the false predictions of our model, in order to analyze the weaknesses of our model. Different samples with false predictions of our Inception model have been depicted in Figure 5.3. As it can be inferred from these examples, most of the samples that the model could not predict correctly, were images of non-informative segments that have been slipped through our filters. The model is uncertain about the prediction of this type of images as both fishing and non-fishing data contain these images to some extent. Therefore the predictions are not correct and these samples are misclassified. However, the number of these images is low, and they can barely affect the performance of the model.

Overall the results indicate that our model can be used on AIS trajectory data for the

task of fishing detection of unseen data. The trained model can predict if the movement of a ship has a similar pattern of the learnt fishing patterns.

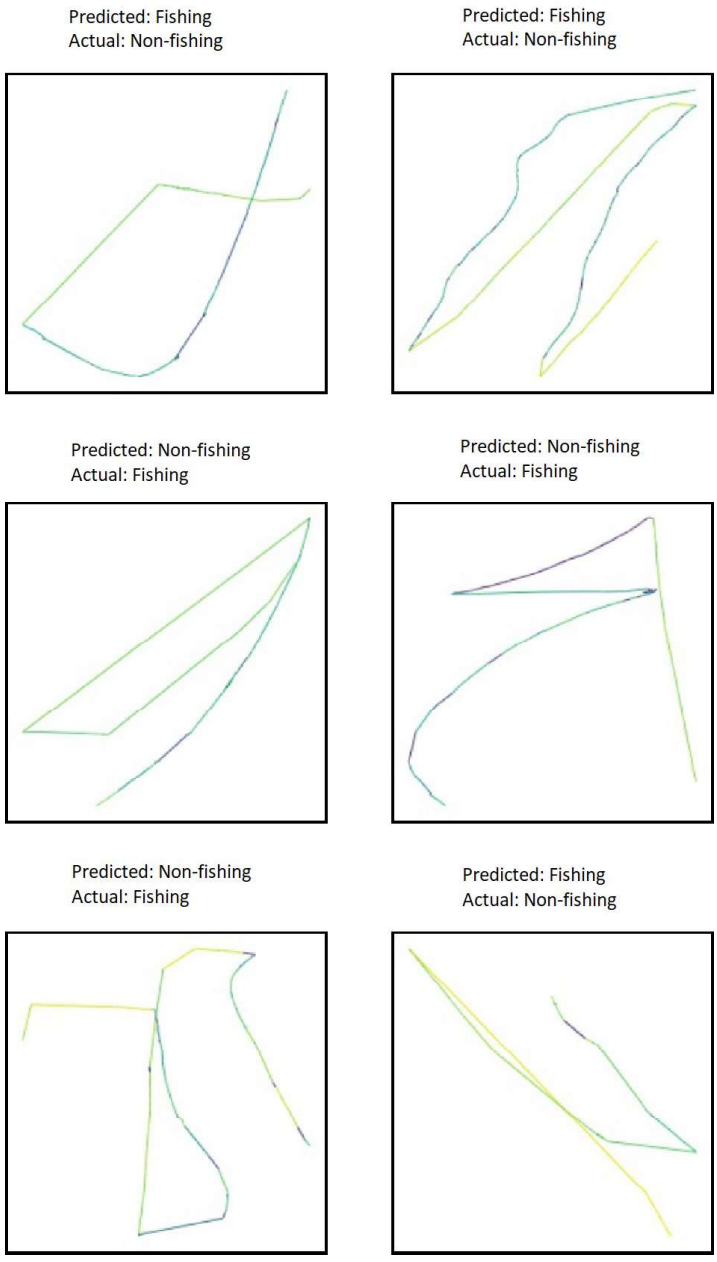



Figure 5.3: Samples that our trained model was unable to classify correctly have been depicted here. The weakness of the model, as it can be seen from these samples, is due to the noise data. These noise samples slip through the filters that have been put in the segmentation part to prevent depicting non-informative segments.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In general, there has been interest in fishing activity detection from trajectory data since the beginning. Patterns of fishing activities have various sizes and forms. Besides, the data that is collected is noisy, and the locations have small errors. Therefore, fishing detection has been a challenge for a long time, and the methods that have been proposed are highly dependent on human aid in the process.

In this thesis, we went over our proposed method for fishing activity detection that eliminates the dependency of the problem on humans. Our proposed method that tackles the fishing detection problem from AIS trajectory dataset, is a new image approach. An image dataset is created from the trajectory data and the model is trained on it. The trajectory dataset that we have is a set of points in time that represent the location of a vessel. With the use of DBSCAN, we separate small informative parts of these trajectories and draw the lines in which the vessel has moved. Thereafter, images are created out of those lines and fed into a CNN. The CNN model is trained to detect fishing patterns from non-fishing ones. The uniqueness of our method is the data transformation part, because we do not feed the raw trajectories into the model.

The trained model is tested on AIS trajectory data from different regions of the world. We show that our results are adequate for a generalized method without having any expert labeled data. In addition, the results show that weak predictions of the model are due to the noisy data, and if the data is cleaned better, the accuracy will improve significantly. Our work opens up a new approach for trajectory related tasks or other data transformation techniques to help solve the problems.

## 6.2 Discussion

In this section, the research questions that were asked in the Introduction chapter are answered.

1. How to detect illegal fishing from AIS data? We tackle this problem by representing the activity segments of the trajectory data as images to classifier. Our method gives satisfactory results for the task that indicates illegal fishing can be detected using our trained model.

2. Can representing trajectories as images improve fishing detection? As it is shown in the Results chapter, the performance of our model gives a good generalized accuracy for the task, which shows the usefulness of this unique approach.

3. How does the performance of our proposed method compare to the other individual state-of-the-art classical approaches? Our approach has a better performance in some metrics than the other models, while for some other, the performance is slightly less satisfying. Although our method gives a better generalized error than training different models for different vessel types.

4. Can our method outperform other individual state-of-the-art classical approaches for all types of vessels? Our method gives satisfactory results in comparison to most of the models. It is worth mentioning that our model is the only method that does not need a labeled cleaned dataset, and all of the steps for the dataset are done automatically without the help of experts.

## 6.3 Limitations

Despite the adequate performance of our model, there are certain limitations to it. These limitations include:

- One of the main limitations of our method, as mentioned in the Results chapter, is the noise images that have been created as fishing images in our dataset. As the number of these images is small, the effect of them on the accuracy of the model is not discouraging. One of the ways that this limitation can be eliminated is by experts.

As a visual analytic task, a human expert can monitor the created images and remove the ones that can affect the learning of the model.

- DBSCAN has its own limitations, and as our method uses it for the segmentation step, these limitations are also present in our model. The main problem with DBSCAN for our model is the elimination of informative images due to the global parameters that do not satisfy all segments.

- As mentioned earlier, the labels for our image dataset come from the ship type code. We assume that if a ship type code does not indicate a fishing vessel, all of the images created from that ship's trajectory are labeled as non-fishing. Therefore the images created from illegal fishing activities of non-fishing vessels are labeled as non-fishing.

## 6.4 Future Work

To the best of our knowledge, our method is the first image approach to solve a trajectory related problem. Therefore, there are many aspects of this approach that can be explored in the future or implemented for similar tasks. Other than that the specific improvements that can be done in future works for our method are as follows:

- By looking at the limitations of our method, it can be understood that the segmentation method and filters need to be improved in the future. The need for global parameters of the segmentation algorithm that produce noise samples or remove useful data should be eliminated.

- One of the ways that the segmentation method can be improved is the automation of finding the parameters based on the time frequency of the ship transmitting points in a trajectory.

- Another future work that can improve our model is using more customized architectures for our CNN. Based on the task and type of data, architectures can be modified to improve the overall performance.

- Exploring active learning algorithms for the task with a visual analytic approach is an interesting future work. The user can help the model while learning to improve

the performance of the model in the segmentation part. Exact borders of the segment can be identified by the user in the procedure to avoid creating noise images.

- The image dataset can be created by depicting the sub-trajectories as time-series. Spatiotemporal data can be converted to time-series in different ways, and images of these series can be saved for classification part in the future.

- Other classical algorithms rather than deep learning models can be tested for the classification of the images dataset in the future.

# Bibliography


[1] Yu Zheng. Trajectory data mining: An overview. *ACM Transaction on Intelligent Systems and Technology*, 2015.

[2] Chire ,Wikimedia Commons. Illustration of en:dbscan cluster analysis. `//upload.wikimedia.org/wikipedia/commons/a/af/DBSCAN-Illustration.svg`, 2011. [Online; accessed 17-October-2019].

[3] Wikimedia Commons. Linear regression. `https://commons.wikimedia.org/wiki/File:Linear_regression.svg`. [Online; accessed 22-October-2019].

[4] Sandra Vieira, Walter HL Pinaya, and Andrea Mechelli. Using deep learning to investigate the neuroimaging correlates of psychiatric and neurological disorders: Methods and applications. *Neuroscience & Biobehavioral Reviews*, 74:58–75, 2017.

[5] Zhenni Feng and Yanmin Zhu. A survey on trajectory data mining: Techniques and applications. *IEEE Access*, 4:2056–2067, 2016.

[6] Yu-Ting Yao and Ren-Hung Hwang. Analysis of Swarm Behavior of Users' GPS Data Based on Boids Model. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, pages 421–427. IEEE, 2014.

[7] Erwin R Gowree, Chetan Jagadeesh, Edward Talboys, Christian Lagemann, and Christoph Brücker. Vortices enable the complex aerobatics of peregrine falcons. *Communications biology*, 1(1):27, 2018.

[8] Konstantinos Chatzikokolakis, Dimitrios Zissis, Giannis Spiliopoulos, and Konstantinos Tserpes. Mining Vessel Trajectory Data for Patterns of Search and Rescue. In *EDBT/ICDT Workshops*, pages 117–124, 2018.

[9] MacDonald, Michael . Hurricane dorian to hit atlantic canada with high winds, downpours. `https://www.cbc.ca/news/canada/nova-scotia/hurricane-dorian-atlantic-canada-eastern-quebec-1.5270554`, 2019. [Online; accessed 17-October-2019].

[10] Erico N de Souza, Kristina Boerder, Stan Matwin, and Boris Worm. Improving fishing pattern detection from satellite AIS using data mining and machine learning. *PloS one*, 11(7):e0158248, 2016.

[11] AIS MESSAGES. `https://navcen.uscg.gov/?pageName=AISMessages`, 2019. [Online; accessed 21-October-2019].

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] Wikipedia contributors. Overfishing — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Overfishing&oldid=917309628`, 2019. [Online; accessed 7-October-2019].

[15] DH Ballard. CM Brown Computer Vision Prentice-Hall. *Englewood Cliffs, New Jersey*, 1982.

[16] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[17] Reinhard Klette. *Concise Computer Vision*. Springer, 2014.

[18] Wen Chen, Minhe Ji, and Jianmei Wang. T-DBSCAN: A spatiotemporal density clustering for GPS trajectory segmentation. *International Journal of Online Engineering (iJOE)*, 10(6):19–24, 2014.

[19] Ting Luo, Xinwei Zheng, Guangluan Xu, Kun Fu, and Wenjuan Ren. An improved DBSCAN algorithm to detect stops in individual trajectories. *ISPRS International Journal of Geo-Information*, 6(3):63, 2017.

[20] Yihong Yuan and Martin Raubal. Measuring similarity of mobile phone user trajectories–a Spatio-temporal Edit Distance method. *International Journal of Geographical Information Science*, 28(3):496–520, 2014.

[21] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 863–868. ACM, 2008.

[22] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, pages 593–604. ACM, 2007.

[23] Sungsoon Hwang, Cynthia VanDeMark, Navdeep Dhatt, Sai V Yalla, and Ryan T Crews. Segmenting human trajectory data by movement states while addressing signal loss and signal noise. *International Journal of Geographical Information Science*, 32(7):1391–1412, 2018.

[24] Jose Antonio MR Rocha, Valéria C Times, Gabriel Oliveira, Luis O Alvares, and Vania Bogorny. DB-SMoT: A direction-based spatio-temporal clustering method. In *2010 5th IEEE international conference intelligent systems*, pages 114–119. IEEE, 2010.

[25] Xianbin Wu, Lin Wu, Yongjun Xu, Zhulin An, and Boyu Diao. Vessel trajectory partitioning based on hierarchical fusion of position data. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 1230–1237. IEEE, 2015.

[26] Gerben Klaas Dirk De Vries and Maarten Van Someren. Machine learning for vessel trajectories using compression, alignments and domain knowledge. *Expert Systems with Applications*, 39(18):13426–13439, 2012.

[27] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on GPS data for web applications. *ACM Transactions on the Web (TWEB)*, 4(1):1, 2010.

[28] Somayeh Dodge, Robert Weibel, and Ehsan Forootan. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, 33(6):419–434, 2009.

[29] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proceedings of the VLDB Endowment*, 1(1):1081–1094, 2008.

[30] Carlos Andres Ferrero, Luis Otavio Alvares, Willian Zalewski, and Vania Bogorny. MOVELETS: exploring relevant subtrajectories for robust trajectory classification. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 849–856. ACM, 2018.

[31] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956. ACM, 2009.

[32] Young-Ji Byon, Baher Abdulhai, and Amer S Shalaby. Impact of sampling rate of GPS-enabled cell phones on mode detection and GIS map matching performance. Technical Report 07-1795, 2007.

[33] Paola A Gonzalez, Jeremy S Weinstein, Sean J Barbeau, Miguel A Labrador, Philip L Winters, Nevine L Georggi, and Roxana Perez. Automating mode detection for travel behaviour analysis by using global positioning systems-enabled mobile phones and neural networks. *IET Intelligent Transport Systems*, 4(1):37–49, 2010.

[34] Ali Yazdizadeh, Zachary Patterson, and Bilal Farooq. Semi-supervised GANs to Infer Travel Modes in GPS Trajectories. *arXiv preprint arXiv:1902.10768*, 2019.

[35] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th International Conference on World Wide Web*, pages 247–256. ACM, 2008.

[36] Theresa Nick, Edmund Coersmeier, Jan Geldmacher, and Juergen Goetze. Classifying means of transportation using mobile sensor data. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2010.

[37] Timothy Sohn, Alex Varshavsky, Anthony LaMarca, Mike Y Chen, Tanzeem Choudhury, Ian Smith, Sunny Consolvo, Jeffrey Hightower, William G Griswold, and Eyal De Lara. Mobility detection using everyday GSM traces. In *International Conference on Ubiquitous Computing*, pages 212–224. Springer, 2006.

[38] Juha Parkka, Miikka Ermes, Panu Korpipaa, Jani Mantyjarvi, Johannes Peltola, and Ilkka Korhonen. Activity classification using realistic data from wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):119–128, 2006.

[39] Arash Jahangiri and Hesham A Rakha. Applying machine learning techniques to transportation mode recognition using mobile phone sensor data. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2406–2417, 2015.

[40] Fabrizio Natale, Maurizio Gibin, Alfredo Alessandrini, Michele Vespe, and Anton Paulrud. Mapping fishing effort through AIS data. *PloS one*, 10(6):e0130746, 2015.

[41] Iraklis Varlamis, Konstantinos Tserpes, and Christos Sardianos. Detecting Search and Rescue missions from AIS data. In *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*, pages 60–65. IEEE, 2018.

[42] Rocío Joo, Sophie Bertrand, Alexis Chaigneau, and Miguel Niquen. Optimization of an artificial neural network for identifying fishing set positions from VMS data: an example from the Peruvian anchovy purse seine fishery. *Ecological Modelling*, 222(4):1048–1059, 2011.

[43] Sophie Bertrand, Erich Díaz, and Matthieu Lengaigne. Patterns in the spatial distribution of Peruvian anchovy (Engraulis ringens) revealed by spatially explicit fishing data. *Progress in Oceanography*, 79(2-4):379–389, 2008.

[44] Rafael Falcon, Rami Abielmona, and Erik Blasch. Behavioral learning of vessel types with fuzzy-rough decision trees. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE, 2014.

[45] Xiang Jiang, Daniel L Silver, Baifan Hu, Erico N de Souza, and Stan Matwin. Fishing activity detection from AIS data using autoencoders. In *Canadian Conference on Artificial Intelligence*, pages 33–39. Springer, 2016.

[46] Marza Ihsan Marzuki, Philippe Gaspar, René Garello, Vincent Kerbaol, and Ronan Fablet. Fishing gear identification from vessel-monitoring-system-based fishing vessel trajectories. *IEEE Journal of Oceanic Engineering*, 43(3):689–699, 2017.

[47] Jing Cao, Maohan Liang, Yan Li, Jinwei Chen, Huanhuan Li, Ryan Wen Liu, and Jingxian Liu. PCA-based hierarchical clustering of AIS trajectories with automatic extraction of clusters. In *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, pages 448–452. IEEE, 2018.

[48] Huanhuan Li, Jingxian Liu, Ryan Liu, Naixue Xiong, Kefeng Wu, and Tai-hoon Kim. A dimensionality reduction-based multi-step clustering method for robust vessel trajectory analysis. *Sensors*, 17(8):1792, 2017.

[49] Xiang Jiang, Erico N de Souza, Xuan Liu, Behrouz Haji Soleimani, Xiaoguang Wang, Daniel L Silver, and Stan Matwin. Partition-wise Recurrent Neural Networks for Point-based AIS Trajectory Classification. In *ESANN*, 2017.

[50] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. Density-based spatial clustering of applications with noise. In *Int. Conf. Knowledge Discovery and Data Mining*, volume 240, 1996.

[51] Mohammed H Al-Shakhs. DAY-AHEAD MARGINAL PRICE FORECASTING OF ELECTRIC POWER SPOT MARKET USING INNOVATED FORECASTING APPROACHES. 2011.

[52] Hilary L Seal. Studies in the History of Probability and Statistics. XV The historical development of the Gauss linear model. *Biometrika*, 54(1-2):1–24, 1967.

[53] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[54] Yuan Yuan and Fuchun Sun. Delay-dependent stability criteria for time-varying delay neural networks in the delta domain. *Neurocomputing*, 125:17–21, 2014.

[55] Nan Hou, Hongli Dong, Zidong Wang, Weijian Ren, and Fuad E Alsaadi. Non-fragile state estimation for discrete Markovian jumping neural networks. *Neurocomputing*, 179:238–245, 2016.

[56] Yajing Yu, Hongli Dong, Zidong Wang, Weijian Ren, and Fuad E Alsaadi. Design of non-fragile state estimators for discrete time-delayed neural networks with parameter uncertainties. *Neurocomputing*, 182:18–24, 2016.

[57] Fan Yang, Hongli Dong, Zidong Wang, Weijian Ren, and Fuad E Alsaadi. A new approach to non-fragile state estimation for continuous neural networks with time-delays. *Neurocomputing*, 197:205–211, 2016.

[58] Jie Zhang, Lifeng Ma, and Yurong Liu. Passivity analysis for discrete-time neural networks with mixed time-delays and randomly occurring quantization effects. *Neurocomputing*, 216:657–665, 2016.

[59] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[60] Li Deng. Three classes of deep learning architectures and their applications: a tutorial survey. *APSIPA Transactions on Signal and Information Processing*, 2012.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[62] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. *URL¡ http://www. deeplearningbook. org*, 2016.

[63] Bottom trawls, technology fact sheets. in: Fao fisheries and aquaculture department. `http://www.fao.org/fishery/geartype/205/en`, 2001. [Online; accessed 17-October-2019].

[64] Patrick E Moffitt. Method and apparatus for long line and recreational bait fishing, August 28 2008. US Patent App. 12/034,192.

[65] Fishing techniques. industrial tuna longlining, technology fact sheets. in: Fao fisheries and aquaculture department. `http://www.fao.org/fishery/fishtech/1010/en`, 2003. [Online; accessed 17-October-2019].

[66] Heather Ball. *Satellite AIS for Dummies*. John Wiley & Sons Incorporated, 2012.

[67] Types of automatic identification systems (per itu-r m.1371 and iec standards). `https://navcen.uscg.gov/?pageName=typesAIS`, 2019. [Online; accessed 21-October-2019].

[68] Fishing for fishermen 2. `https://community.topcoder.com/longcontest/?module=ViewProblemStatement&rd=16978&pm=14691`, 2019. [Online; accessed 13-November-2019].

[69] Kevin P Murphy. *Machine learning: a Probabilistic Perspective*. MIT press, 2012.

# Appendix A

## Programming details

All of the code has been written in Python 3.7.0, and the following libraries have been used:

1. **Anaconda Python:** We have used Anaconda version 2019.03 for the installation and easy setting of our environment. Anaconda is an open-source distribution that manages the dependencies and prepares most used Data Science and Machine Learning libraries. Many libraries that we have used, including Matplotlib, are installed using Anaconda.

2. **Scikit-learn:** For the segmentation part of our model, the DBSCAN algorithm from scikit-learn library version 0.21.2, have been used. This library also provides different classification and evaluation functions.

3. **Matplotlib:** One of the main parts of our method is the image creation part that has been done using Matplotlib version 3.1.0. This library is a plotting library that can draw the wanted patterns and save them as images.

4. **PyTorch:** PyTorch version 1.1.1 developed by Facebook's artificial intelligence research group, is a machine learning library. PyTorch is used in our code for implementing the CNN architectures, and all of the training and testing steps of them.

5. **SciPy:** For our scientific computing purposes, we have used SciPy version 1.2.0. This library provides us with a fast implementation of linear regression for our filtering steps.